UNIVERSIDAD POLITÉCNICA DE CARTAGENA
Escuela Técnica Superior de Ingeniería Naval y Oceánica

Splines Cúbicos Suavizantes en el Diseño Naval

Autor: Blanca Roncero Peña
Director: Juan Carlos Trillo Moya

Departamento de Matemática Aplicada y Estadística

Splines Cúbicos Suavizantes en el Diseño Naval

(Cubic Smoothing Splines in Naval Design)

Proyecto Fin de Carrera

Blanca Roncero Peña

Ingeniería Naval y Oceánica

Agradecimientos

Al Doctor D. Juan Carlos Trillo Moya, por compartir conmigo todo su conocimiento y la gran ayuda prestada durante el desarrollo de este proyecto fin de carrera.

Al Departamento de Matemática Aplicada y Estadística y a la Universidad Politécnica de Cartagena.

A mis padres, por todo su esfuerzo y sacrificio, y a mis hermanos y a mi abuela por estar ahí.

A mis amigos, especialmente a mi gran amigo Ángel quien me ha prestado su apoyo incondicional este último año.

A Francisco, por darme todo su apoyo y compartir conmigo la ilusión por ser Ingeniera.

Sinopsis

Durante los últimos años, se han dedicado muchos esfuerzos para desarrollar las matemáticas de las funciones spline. Gran parte de este interés es debido a la alta importancia que tienen los splines en el diseño industrial. Estas funciones spline han encontrado una amplia aplicación, principalmente en el campo de la interpolación. Sin embargo en los últimos años ha habido una alta demanda para reemplazar la interpolación estricta por algún tipo de suavizado. Este suavizado normalmente es preferido si los valores de las ordenadas se dan sólo de forma aproximada, por ejemplo, si estos datos provienen de una fuente experimental la cual puede introducir cierto grado de ruido.

Este documento presenta el algoritmo de los splines cúbicos suavizantes junto con su implementación en Matlab mediante una interfaz gráfica. Esta función spline ajusta una función cúbica suavizante a la información proporcionada por el usuario, la cual se verá influenciada por unos coeficientes de peso capaces de controlar el grado de suavizamiento de la curva. Asimismo, a lo largo de este proyecto se justificará el uso de splines en el diseño naval, los cuales ganan importancia a la hora de obtener ''carenas lisas'', es decir sin abolladuras, pliegues o arrugas, las cuales causarían perturbaciones en el flujo que la atraviesa, el cual conlleva un aumento de la resistencia al avance, mayor consumo de combustible y una notable pérdida de eficiencia del buque.

Comenzaremos dicho proyecto enunciando unas nociones básicas sobre diseño naval, fundamental para poder entender la importancia de los splines tanto en el diseño manual como asistido por ordenador. A continuación se expondrá de forma detallada la obtención del algoritmo de cálculo para los Splines Cúbicos Suavizantes seguido del desarrollo de una interfaz gráfica mediante la función GUI de Matlab, programa de desarrollo matemático muy usado en el campo de la ingeniería. Al ser un proyecto de final de carrera de la titulación de Ingeniería Naval y Oceánica, en los últimos capítulos se tratará de resolver un caso práctico referente al diseño naval.

Abstract

During past years, many efforts have been dedicated to develop mathematical spline functions. Much of this interest is due to the high importance of splines in industrial design. These functions have found wide application, particularly in the field of interpolation. However in recent years there has been a high demand to replace the strict interpolation by some type of smoothing. This smoothing is usually searched for when ordinate values are given only approximately, for example, if the data is coming from an experiment which may introduce some degree of noise.

This paper presents an algorithm of cubic smoothing splines with its implementation in Matlab using a graphical interface. This function fits a cubic spline to the information provided by the user, which will be influenced by weight coefficients capable of controlling the degree of smoothing of the curve. Also throughout this project the use of splines will be justified in naval design, which gains importance in obtaining ''smooth hulls'', that are the same as a hull without dents, creases or wrinkles, which may cause disturbances in the flow there through, which entails an increase in the forward resistance, increasing the fuel consumption and losing efficiency of the vessel.

This project will begin by enunciating a basic understanding of naval design, essential to understand the importance of splines in both manual and computer assisted design (CAD). Next it will detail how the algorithm for cubic smoothing splines is obtained followed by the development of a graphical interface using the Matlab GUI function, which is a mathematical development software widely used in the field of engineering. As a final thesis of the degree in Naval and Oceanic Engineering, in the last chapters it will solve a practical case which is related to naval design.

Índice de Contenidos

Índice de Figuras

Capítulo I

Capítulo II

Capítulo III

Capítulo IV

Capítulo V

Capítulo VI

Capítulo VII

Objetivos

El objetivo del presente Proyecto Fin de Carrera consistirá en desarrollar y estudiar la función Spline Cúbico Suavizante, que podrá utilizarse para realizar aproximaciones y obtener curvas suaves las cuales pueden ser utilizadas en programas de diseño industrial mediante ordenador, incluyéndose en el campo del dibujo naval.

En este proyecto:

- Se estudiará la necesidad de las curvas suaves en el diseño naval, se describirá el proceso de diseño de las formas de una embarcación tanto manual como mediante el ordenador.
- Se describirán los procesos de alisado de las formas de una embarcación, procesos necesarios e imprescindibles de conocer para todo diseñador de estructuras navales.
- Se estudiará la función spline, asimismo se comparará con otros métodos de interpolación, demostrando las ventajas que éstos presentan.
- Se desarrollará un algoritmo de cálculo para la función splines cúbicos suavizantes.
- Este algoritmo será desarrollado en código M o Matlab, herramienta muy potente en el campo de las matemáticas.
- Se desarrollará una interfaz gráfica capaz de ejecutar este algoritmo de cálculo de splines cúbicos suavizantes, permitiendo el uso de esta herramienta a cualquier tipo de usuario.
- Será demostrada la eficiencia del algoritmo desarrollado mediante un caso práctico perteneciente al campo del diseño naval.

Capítulo I
Introducción al Diseño Naval

Capítulo I

Introducción al Diseño Naval

Para poder entender el proceso de diseño de una embarcación, es importante conocer previamente las partes y características que definen a un buque, así como conocer la nomenclatura específica para poder posicionarse en cualquier parte de una embarcación.

En los siguientes apartados del presente proyecto, se va a tratar de proporcionar al lector unas nociones generales de las partes o zonas de referencia que le serán de utilidad para definir una embarcación.

1.1 Conceptos básicos.

Los conceptos que van a ser descritos en este apartado, son bastante básicos en el ámbito de la construcción naval, sin embargo es necesario conocer cada uno de ellos para poder comprender el proceso de diseño de una embarcación.

Las partes principales de un barco junto con los términos que se aplican a estas partes, son *proa (bow) y popa (stern)* como las partes delantera y trasera del buque en el sentido normal del movimiento, *estribor (starboard side) y babor (port side)* como las bandas o costados del buque que quedan respectivamente a la derecha y a la izquierda de un observador que mirase en el sentido proa-popa, y las *amuras (port bow, star bow) y aletas (port quarter, starboard quarter)* como las zonas de los costados de proa y popa del buque respectivamente.

Un buque consta esencialmente de una caja estanca de forma adecuada a su función, llamada casco, sobre la cual se erige una superestructura. Parte del casco está sumergida constituyendo la *obra viva* o *carena*, y el resto emerge llamándose *obra muerta*.

La superficie del casco está formada por chapas de acero, tablas de madera u otro material adecuado de un cierto espesor, pero a fin de representarlo convenientemente, sin ambigüedad, empezaremos por imaginar un casco ideal de espesor infinitamente pequeño, es decir, una superficie geométrica con respecto a la cual situaremos los elementos estructurales. Esta superficie ideal o de diseño, tiene normalmente un solo plano de simetría longitudinal llamado crujía. El plano de crujía, es por su importancia, uno de los planos básicos de referencia en la representación de la superficie de diseño

Si se realizara un corte transversal al buque, quedarían definidas las siguientes partes del mismo tal y como se indica en la figura siguiente:

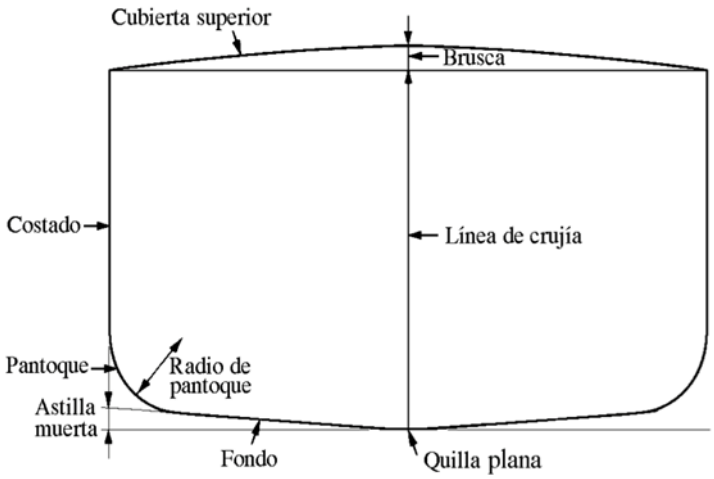


Figura 1.1: Corte transversal de una embarcación.

En la figura 1.1 quedan definidas las partes más básicas al realizar un corte transversal al buque. Se define la *cubierta superior* como la superficie de cierre superior del casco y la *brusca* es la curvatura transversal de la cubierta medida por la altura de la cuerda en crujía, desde la cara inferior de la cubierta hasta el punto más alto del costado.

En los laterales de la figura se aprecian los *costados* como cada uno de los laterales del casco situados entre el pantoque y la cubierta superior. El *pantoque* es la zona curva de unión entre el fondo y el costado del barco.

En la parte más baja de la figura se definen la *quilla plana,* como la zona inferior y en crujía del forro del casco, y el *fondo*, que es la parte inferior del casco junto a la quilla. Por último, la *astilla muerta* es definida como la elevación de la cuaderna sobre el plano base.

La nomenclatura más importante que recibe el casco visto longitudinalmente es la que puede apreciarse en la figura 1.2. En ella queda definida la *roda,* como la zona más a proa del casco donde se unen los costados, el *codaste*, como la zona más a popa del casco donde se unen los costados por debajo de la flotación, y por último el *arrufo* de cubierta o también la curvatura estructural que se le da a la cubierta en sentido longitudinal. El arrufo varía a lo largo de la eslora siendo mayor en los extremos.

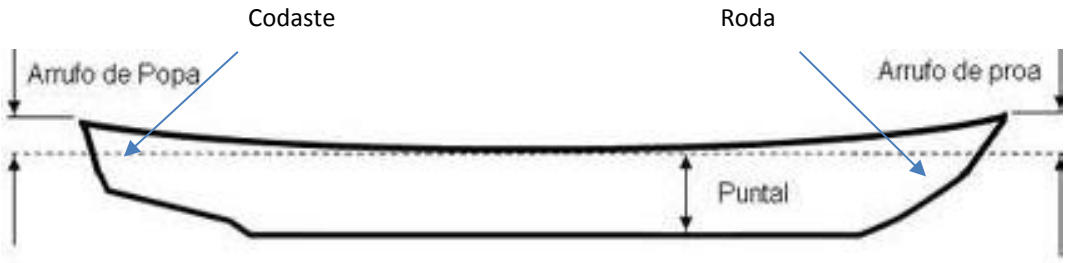


Figura 1.2: Vista longitudinal de una embarcación.

1.2 Dimensiones Principales del Buque.

Las dimensiones principales que definen a un buque son la eslora, la manga, el puntal y el calado. A continuación se va a detallar en que consiste cada una de ellas, así como sus tipos.

La *eslora (length)* es la dimensión del barco en sentido longitudinal, es decir, de proa a popa. A menudo las formas de éstas suelen ser complicadas, siendo necesario definir varias esloras.

La manga *(breadth)* es la dimensión de barco en sentido transversal. Se puede distinguir entre las siguientes mangas:

-La *manga de trazado* será la máxima dimensión transversal de trazado del casco del buque a lo largo de la eslora.

-La *manga fuera de forros* es la manga de trazado sumándole los espesores de las dos planchas del forro.

-La *manga en una flotación* es la manga máxima en la flotación considerada.

El *puntal* (depth) es la dimensión en sentido vertical del buque. Se pueden considerar los siguientes puntales:

-El *puntal de trazado o de construcción* es la distancia vertical medida en el centro del buque, desde la cara alta de la quilla a la cara alta del bao (refuerzo estructural de la cubierta en el sentido transversal).

El *calado (draught)* es la distancia vertical correspondiente a la parte sumergida del buque. Se pueden considerar los siguientes calados:

-El *calado de trazado* es la distancia vertical de trazado de la parte sumergida del casco del buque por debajo de la flotación de trazado o proyecto, medida en la sección media.

- El *calado en una flotación* o calado real en esa flotación es el calado medido desde la cara inferior o exterior de la quilla hasta el nivel de la flotación correspondiente.

Los **calados a proa y popa** se representan por T_{Pr} y T_{Pp} y son los calados reales del buque en las perpendiculares de proa y popa respectivamente, que se trazan en las intersecciones de la línea de la flotación de proyecto con la roda y el codaste. El *calado en la sección media* será el correspondiente en la sección media del buque, sin embargo el *calado medio* se define como la semisuma de los calados a proa y popa.

1.3. Definición del casco de una embarcación.

Se llama *casco* del buque al conjunto estructural del mismo formado por el forro exterior estanco y los refuerzos sobre los que se apoya. *Estanco* significa que es impermeable, es decir, no deja pasar el agua. El casco de un buque se puede cortar según un conjunto de tres planos perpendiculares entre sí, paralelos a los planos de un triedro.

1.3.1 Planos y líneas de referencia del casco.

Para la representación y definición de las formas de un buque es necesario considerar un sistema de referencia tridimensional ortogonal asociado al mismo como el que puede verse en la figura 1.3.

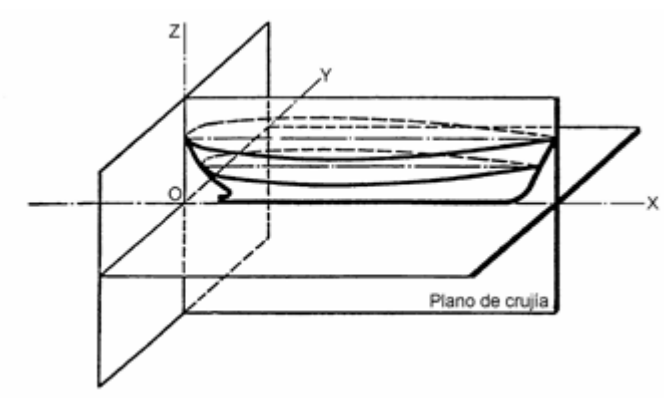


Figura 1.3: Planos de referencia.

En la imagen anterior se han visto los diferentes planos de referencia del buque, así el plano de formas, que será descrito en los próximos apartados, mostrará la forma del buque al referirlo en estos planos.

La superficie exterior de un buque se representa por medio de curvas de nivel equidistantes y paralelas a los planos de referencia, análogamente al procedimiento utilizado en Topografía y, por lo tanto, bastaría definir teóricamente una serie de curvas.

En el caso de la carena de un buque el empleo de una sola serie de curvas no permite calcular con suficiente precisión los diversos elementos de la carena (volúmenes, centros de gravedad, etc.), de las que depende la flotabilidad y estabilidad. Tampoco nos permite comprobar la perfecta continuidad de la superficie de carena, continuidad absolutamente indispensable para reducir al máximo la resistencia al avance y poder garantizar la estanqueidad del casco. Por esta razón se representan las carenas de las embarcaciones por medio de tres series de ''curvas de nivel'', paralelas a los tres planos de referencia, combinadas con secciones oblicuas, si son necesarias. En resumen sabemos que la concordancia de las tres proyecciones nos garantizará la continuidad de las formas de una embarcación.

La superficie que se representa es la exterior, es decir, *fuera de forros*, si los planos se trazan con arreglo a los procedimientos franceses, la superficie fuera de miembros, si se sigue la práctica inglesa; los planos españoles se hacen fuera de miembros.

Los ***planos de referencia*** que nos ayudan a definir las formas de una embarcación son los tres siguientes:

-*Plano de crujía*: es el plano de simetría del barco en sentido longitudinal. La intersección de este plano con el casco se llama línea de crujía. Los planos paralelos al de crujía que cortan al casco del buque se llaman planos longitudinales, y a las líneas de corte de los mismos con el casco, se les llaman longitudinales. (Sheer plan)

-*Plano de flotación*: es el plano perpendicular al de crujía que representa la superficie del agua sin oleaje. El plano de flotación de trazado, o flotación de trazado, es el situado al calado de trazado o proyecto del buque. La intersección de este plano con el casco se llama *línea de flotación de trazado*. Las intersecciones de planos paralelos al de flotación con el casco se llaman *líneas de agua*. Se llama plano base al plano paralelo a la flotación de trazado que pasa por el canto superior de la quilla en la sección media. A la intersección del plano base con el de crujía se le llama *línea base*. (Breadth Plan)

-*Plano transversal*: es un plano perpendicular a los dos anteriores. Las intersecciones de planos transversales con el casco se llaman *cuadernas de trazado o secciones*. (Body Plan)

Para completar la representación de la carena, principalmente la de los fondos, se hace uso de secciones oblicuas que se llaman *vagras planas (diagonales)*. Estas curvas son las intersecciones de la superficie de carena con planos perpendiculares al transversal, elegidos de modo que sean normales a la carena si es posible, en las partes más importantes de dicha superficie.

Las formas del casco de un buque se representan mediante un conjunto de líneas de trazado según los tres tipos de planos anteriores, es decir, *cuadernas, líneas de agua, longitudinales y diagonales o vagras,* como podemos ver en la figura 1.4:

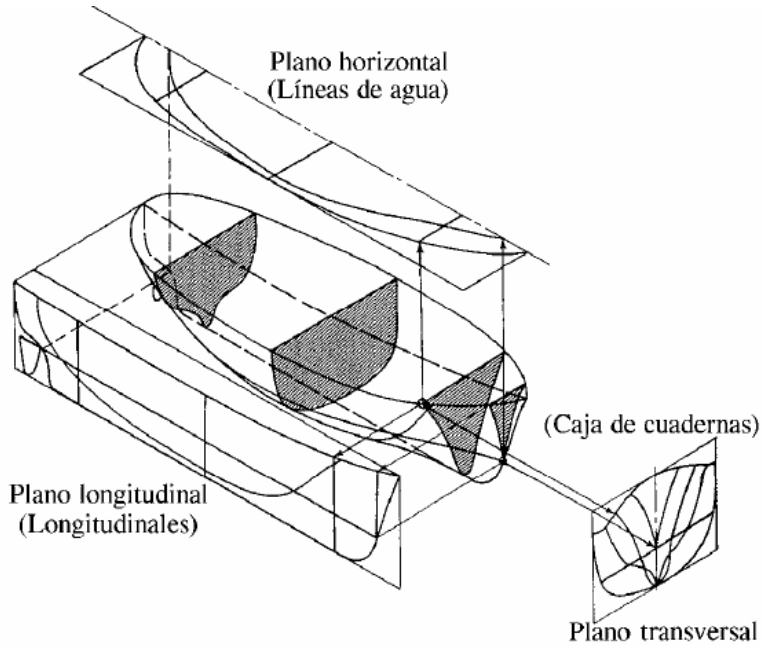


Figura 1.4: Planos que definen una embarcación.

El conjunto de todas estas líneas, serán las que definan las formas de nuestro buque, y éstas, como ya se comentó anteriormente irán contenidas en el denominado **plano de formas**.

1.3.2 La cartilla de trazado.

Las formas de un barco quedan definidas mediante *la cartilla de trazado* que son una serie de cifras que representan puntos en el espacio consistentes en las semimangas de cada sección que corresponden a la intersección sobre las líneas de agua o sobre diagonales que definen mejor los puntos en el plano transversal.

Secciones

	C0	C1/2	C1	C1 1/2	C2	C3	C4	C5	C6	C7	C8	C9	C10
L.A. 0	----------	----------	----------	----------	0,15	0,46	0,6	0,2	----------	----------	----------	----------	----------
L.A. 1	----------	----------	----------	----------	0,46	1,91	3,11	3,16	2,49	1,61	0,83	0,31	----------
L.A. 2	----------	----------	----------	0,48	1,49	3,28	4,13	4,18	3,68	2,74	1,66	0,7	----------
L.A. 3	----------	----------	0,38	1,64	2,83	4,14	4,62	4,66	4,22	3,3	2,11	0,94	----------
L.A. 4	----------	----------	1,63	3,05	3,85	4,58	4,87	4,91	4,54	3,64	2,42	1,12	----------
L.A. 5	----------	1,92	3,28	3,98	4,43	4,81	4,98	5	4,72	3,9	2,66	1,27	----------
L.A. 6	----------	3,39	4,04	4,43	4,71	4,93	5,02	5,03	4,84	4,11	2,88	1,43	0
L.A. 7	----------	4,02	4,44	4,7	4,87	5,01	5,05	5,05	4,92	4,28	3,11	1,6	0,06
L.A. 8	----------	4,28	4,61	4,84	4,97	5,04	5,06	5,07	4,98	4,43	3,34	1,8	0,13
L.A. 9	----------	4,39	4,69	4,9	5,03	5,06	5,07	5,1	5,02	4,59	3,6	2,04	0,22
L.A. 10	3,99	4,46	4,76	4,94	5,05	5,09	5,1	5,11	5,05	4,76	3,89	2,32	0,35
Alt. Cub.	5,7	5,7	5,7	5,7	5,8	5,85	6,05	6,2	6,4	6,5	6,87	7,4	8,05
Sem. Cub.	4,02	4,48	4,78	4,95	5,06	5,1	5,12	5,12	5,1	5,05	4,7	3,77	1,62
Alt. Pie C.	----------	2,235	1,448	0,635	-0,3	-0,2	-0,1	0	0,13	0,225	0,32	0,4	0,33
Sem. Pie C.	----------	0	0	0	0,1	0,2	0,2	0,2	0,2	0,2	0,15	0	0

Figura 1.5: Ejemplo de una cartilla de trazado.

1.4. Representación del casco de una embarcación. El plano de formas.

Como ya venimos comentando, un plano de formas consta de un conjunto de líneas representadas en tres proyecciones ortogonales. La representación de estas líneas va a ser detallada más específicamente a continuación.

La ***Caja de Cuadernas*** o la proyección en un plano transversal, conteniendo a los ejes OY y OZ es la representación del conjunto de todas las cuadernas o secciones de trazado dispuesta en una vista del plano que se denomina caja de cuadernas, situándose las secciones de proa a la derecha de crujía, y las de popa a la izquierda.

Para el trazado de la caja de cuadernas se dividirá la eslora del buque en 10 partes iguales (11 secciones) si el buque es pequeño y en 20 (21 secciones) en buques medianos o grandes. Por el mismo motivo, en las zonas extremas, al ser mayores las curvaturas, se trazan

secciones auxiliares cuya separación son 1/4 ó 1/2 del intervalo normal. La numeración de las secciones tiene el origen en la perpendicular de popa en Europa y en la de proa en EEUU.

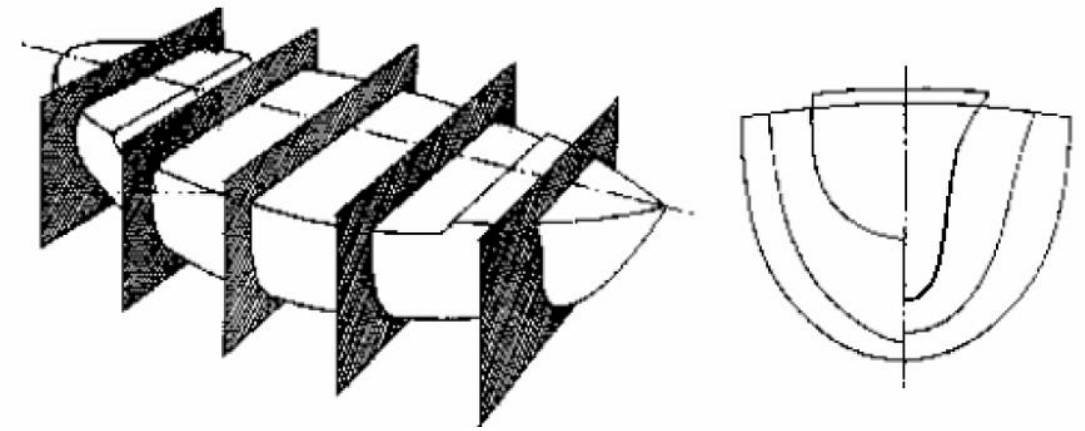


Figura 1.6: Proyección en el plano transversal.

Las *líneas de agua o la proyección en un plano horizontal*, conteniendo a los ejes OX y OY debido a la simetría del barco solo son representadas en la parte de babor. La proa se representa a la derecha.

La línea de agua cero será la correspondiente al plano base y la línea de agua 6 se hace coincidir con la flotación de trazado, por lo que basta dividir el calado en seis partes iguales para obtener la separación entre ellas. En barcos de mayor calado se divide en diez partes en vez de seis. En la zona más baja del buque se suele representar la línea de agua 1/2. Las L.A. (líneas de agua) por encima de la flotación quedan a juicio del proyectista como se muestra en la figura 1.7. (En este caso solo hemos mostrado 6 L.A, aunque lo habitual es mostrar 6).

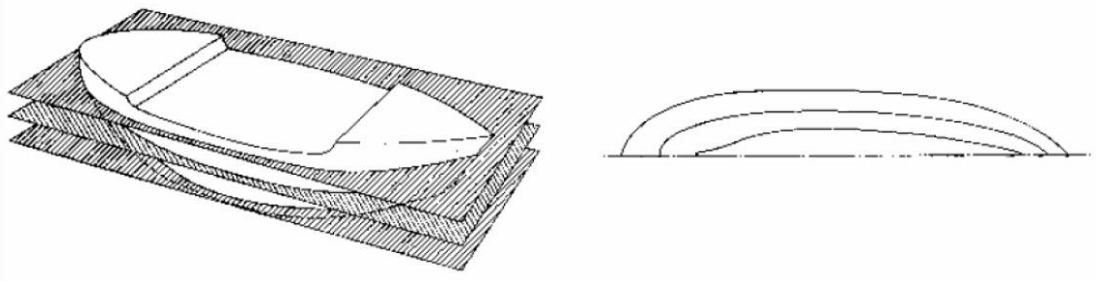


Figura 1.7: Proyección en el plano horizontal.

Las líneas longitudinales o la proyección en un plano longitudinal, conteniendo a los ejes OX y OZ contendrán los diferentes longitudinales, como vemos en la figura 1.8.

Éstos son equidistantes y suelen ser 3 o 4 sin contar crujía, aunque esta norma es flexible. Se representan por números romanos. La proa se representa a la derecha.

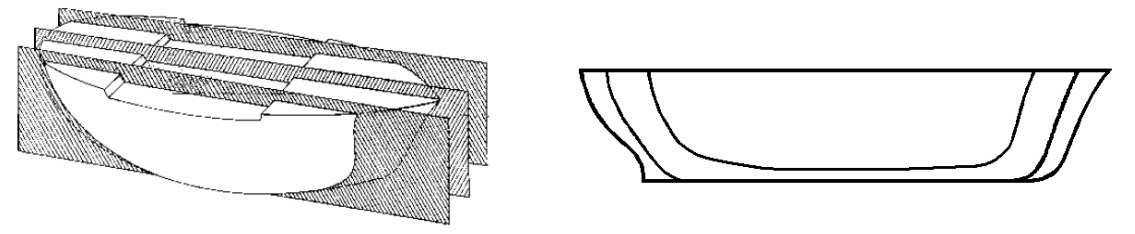


Figura 1.8: Proyección en el plano longitudinal.

Las *vagras o diagonales* serán las resultantes de cortar el buque por planos perpendiculares al transversal y que forman un ángulo determinado con el plano de crujía. Se representan en la caja de cuadernas.

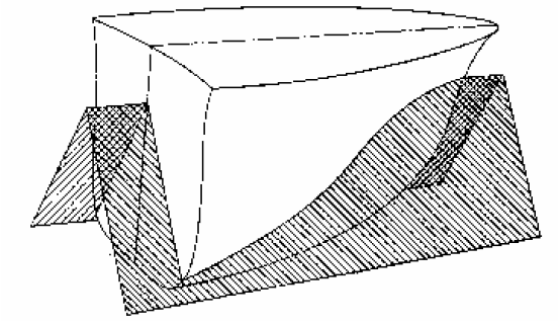


Figura 1.9: Proyección de planos perpendiculares.

La *línea de crujía* es la intersección del casco con el plano diametral o de crujía. *La línea base* es la intersección del plano base con el de crujía, siendo el plano base el plano horizontal que pasa por el canto superior de la quilla en la sección media.

No obstante a todo lo anterior, *no hay una normalización fija para todos los barcos*, en cuanto al número de líneas de agua, secciones y cortes longitudinales a representar en un plano de formas. La *escala de dibujo* siempre será una de las normalizadas (1/25, 1/50, 1/75, 1/100, ...).

Por último la caja de cuadernas se puede colocar en el centro de la proyección longitudinal, en el caso de barcos con cuerpo cilíndrico ya que no resta información sobre las formas. En caso contrario, la caja de cuadernas se colocará a la derecha de la proyección longitudinal.

En los cascos metálicos el trazado se realiza *fuera de miembros*, mientras que en barcos no metálicos, donde el espesor del forro es mayor y por tanto podría dar lugar a errores importantes, se realiza *fuera de forros*.

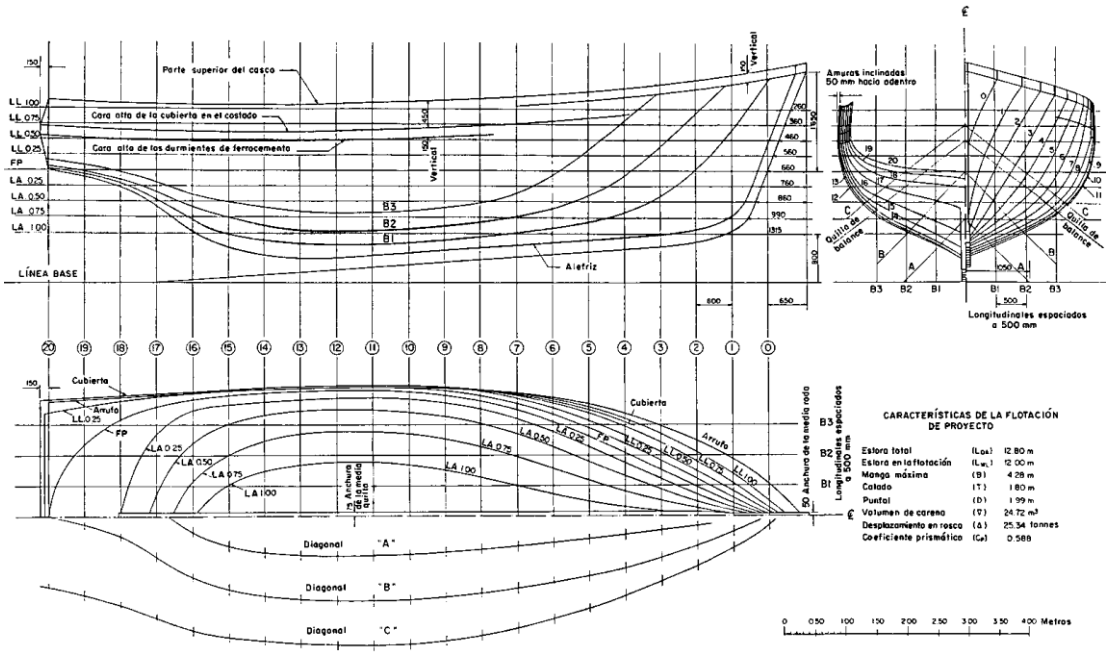


Figura 1.10: Plano de formas.

1.4.1 Representación manual del plano de formas. Necesidad de las curvas suaves.

El dibujo manual de las líneas del plano de formas se realiza mediante el empleo de reglas de gran tamaño, plantillas, y junquillos que se mantienen en posición mediante pesas para trazar las líneas curvas de gran curvatura.

Hoy en día este proceso de dibujo del plano de formas se ha visto simplificado gracias a la aparición de programas de dibujo tales como Rhinoceros, Autocad, Maxsurf, Solidworks… Estos programas permiten al usuario trabajar en tres dimensiones con la embarcación, pudiendo obtener todas las líneas que conforman el plano de formas mediante intersecciones con los planos de referencia. Podría de interés el explicar cómo y por qué se usan los splines o junquillos en el diseño naval, y eso podrá ser comprendido a través de la explicación del proceso de diseño del plano de formas de forma manual.

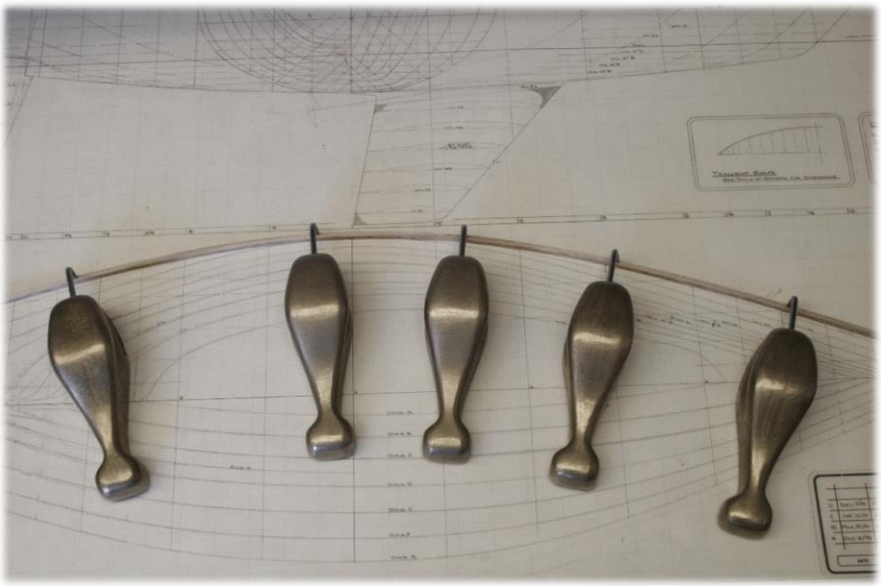


Figura 1.11: Junquillo y pesas (splines and weights).

Partiendo de la cartilla de trazado y de los croquis de proa y popa, se procede a dibujar el plano de formas en varias etapas que podemos resumir de la siguiente manera:

Una primera etapa donde se dibujan las trazas del plano en las tres proyecciones, tal y como se muestra en la figura siguiente.

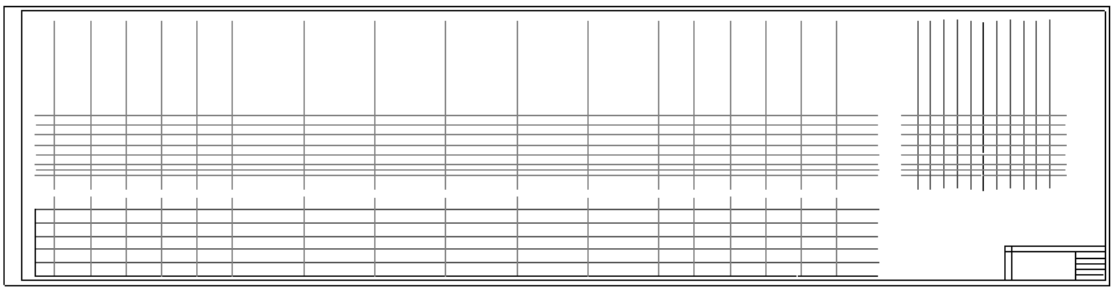


Figura 1.12: Primera etapa en el dibujo del plano de formas.

En la segunda etapa se dibuja la caja de cuadernas a partir de las semimangas obtenidas de la cartilla de trazado. Las cuadernas de proa se representarán a la derecha de crujía, y las de popa a la izquierda. Para representar una cuaderna, se toman las cifras contenidas en la columna correspondiente de la cartilla, pues representan semimangas. La coordenada vertical que corresponde a cada semimanga es la altura de la línea de agua correspondiente, el resultado obtenido puede verse en la siguiente figura.

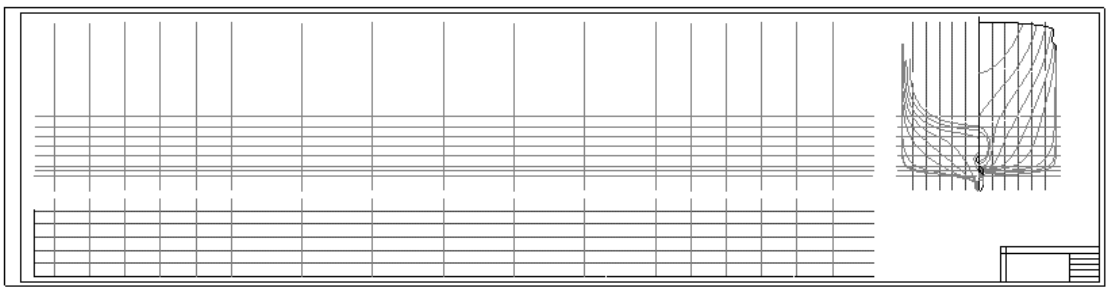


Figura 1.13: Segunda etapa en el dibujo del plano de formas.

En la tercera etapa se representará el perfil del casco en la proyección longitudinal. Para ello se utilizan las alturas de los pies de cuaderna y de cubierta obtenidos de la cartilla de trazado para cada una de las secciones, así como los croquis de proa y popa, como vemos en la figura 1.14:

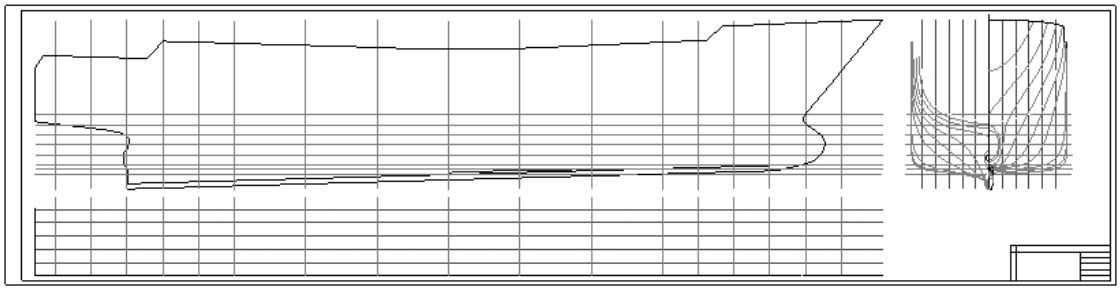


Figura 1.14: Tercera etapa en el dibujo del plano de formas.

A continuación en la cuarta etapa se dibujan las líneas de agua y el perfil de cubierta en la proyección horizontal. Para ello se emplean las semimangas de la cartilla de trazado, pero ahora en el sentido horizontal. Los extremos de proa y popa donde terminan las líneas de agua se obtienen de la proyección longitudinal, en los puntos de corte de la traza correspondiente a cada línea de agua con el perfil longitudinal.

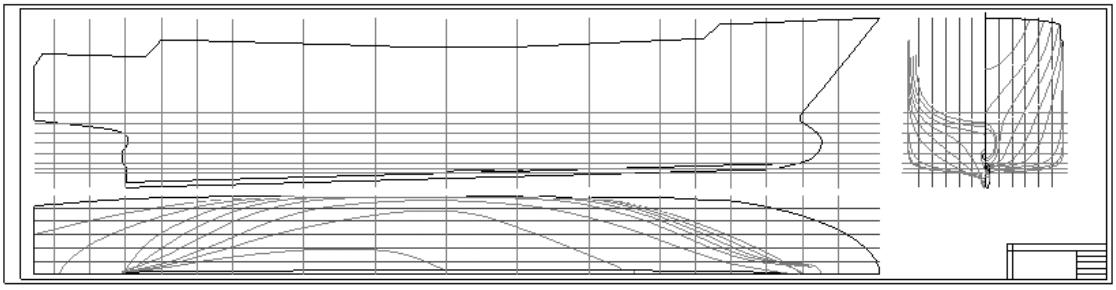


Figura 1.15: Cuarta etapa en el dibujo del plano de formas.

En la **Quinta Etapa** se obtienen los longitudinales prescindiendo ya de la cartilla de trazado, a partir de lo ya dibujado, mediante la proyección sobre la vista longitudinal de los puntos de corte de la traza del longitudinal que se quiere dibujar, con las secciones a las que corta en la caja de cuadernas, y con las líneas de agua a las que corta en la proyección horizontal. Esto queda ilustrado en la siguiente figura:

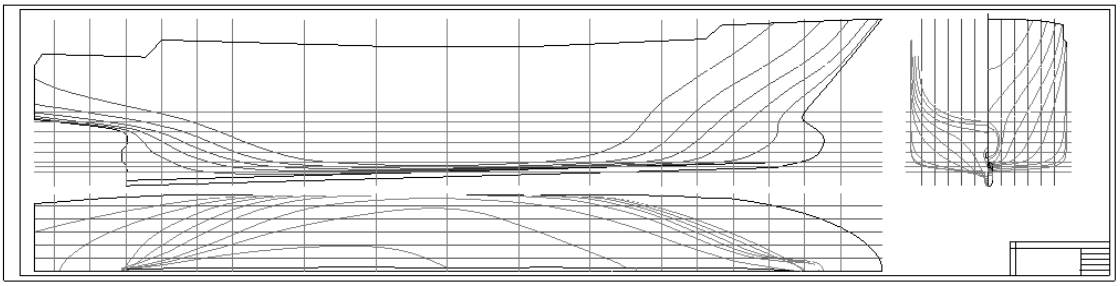


Figura 1.16: Quinta etapa en el dibujo del plano de formas.

Si el longitudinal presenta alguna abolladura, debe procederse a su ***alisado***, proceso en el que intervienen las tres proyecciones ya que una modificación en una vista provocará variaciones en las otras dos. En representación manual este proceso da lugar a deterioro del papel. En las figuras siguientes se representa un ejemplo sencillo de alisado de un longitudinal.

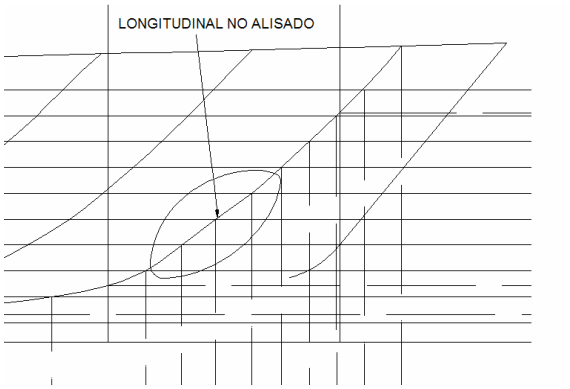


Figura 1.17: Longitudinal no alisado.

Para proceder al alisado se prescinde de los puntos que provocan la abolladura dejando que el junquillo tome una forma no forzada en esa zona. Esto provoca nuevos puntos de corte del longitudinal con las L.A. próximas, debiendo proceder a llevar estos puntos de corte a la proyección horizontal y modificar y alisar las líneas de agua afectadas. Han de corregirse todas aquellas modificaciones que dicho alisado pudiese provocar, de modo que exista correspondencia entre las tres vistas, como podemos apreciar en las siguientes figuras.

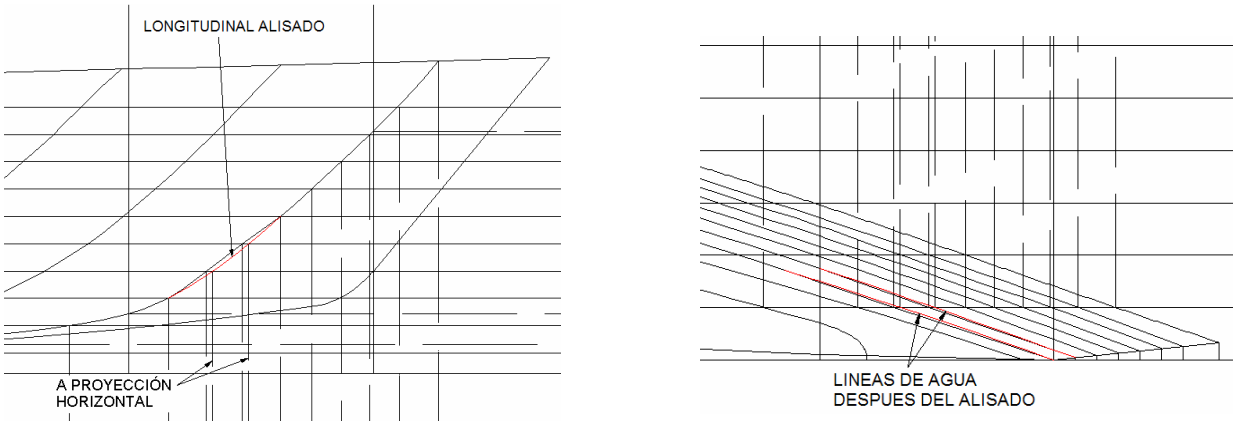


Figura 1.18: Alisado.

Por último en la sexta etapa se realiza el rotulado del plano de formas.

Las cuadernas se rotulan en la caja de cuadernas, y en la parte inferior de las trazas en las otras dos vistas, comenzando por la sección 0 en la perpendicular de popa.

Las líneas de agua se rotulan en la proyección horizontal, a izquierda y derecha, y en las trazas de las otras dos vistas, a izquierda, derecha y entre la proyección longitudinal y la caja de cuadernas. La línea de agua cero es la línea base.

Los longitudinales se rotulan en la proyección longitudinal, a izquierda y derecha, así como en la parte inferior de las trazas en la caja de cuadernas, y a izquierda y derecha de las trazas en la proyección horizontal. Se enumeran con números romanos empezando con el I por el más cercano a crujía.

También se incluyen las *dimensiones principales del barco* en la parte derecha tales como la eslora, manga, puntal y calado como vemos en la siguiente figura:

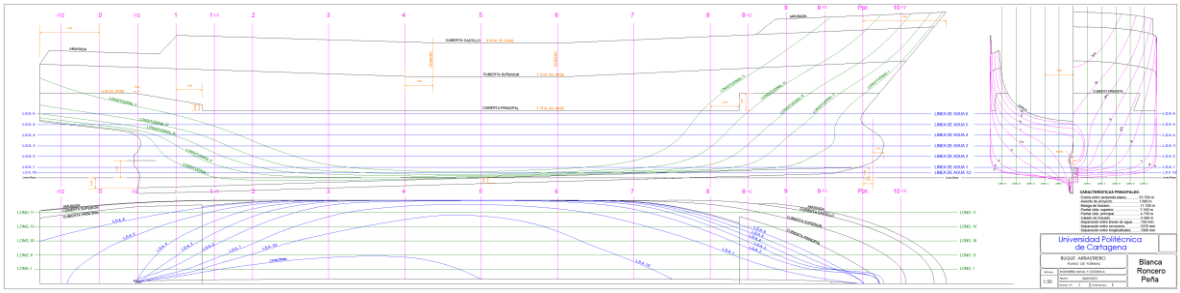


Figura 1.19: Séptima etapa en el dibujo del plano de formas.

Llegados a este punto, el lector habrá podido comprender el porqué de la necesidad de las líneas curvas suaves en el ámbito de la construcción naval, así como la necesidad del alisado de las formas y por qué surge.

Este proceso será descrito con más detalles en el próximo capítulo.

Capítulo II

Alisado y Armonizado de Formas

Capítulo II

Alisado y Armonizado de Formas

En el capítulo anterior ya hemos tratado como se diseñan los planos de construcción de un buque, sin embargo los trabajos han de realizarse con mucha precisión y las formas del buque, como ya se sabe, son tan poco geométricas, que la simple medición sobre el plano con escalas en general de 1:50 como mínimo produciría grandes errores, y en consecuencia esto haría imposible la construcción del buque.

Debido a esto, no hay más remedio que efectuar un trazado de mayor precisión que permita desarrollar con suficiente exactitud los elementos de las estructuras.

Desde hace un tiempo el problema de desarrollar matemáticamente la superficie del casco ha merecido la atención de los investigadores que han tratado de encontrar ecuaciones matemáticas que definan las líneas para poderlas aplicar al cálculo hidrodinámico. En la actualidad este interés se ve aumentado debido a la presión de los astilleros y diseñadores que tratan de encontrar una solución matemática a los problemas de trazado y desarrollo que permita trabajar con mayor precisión y rapidez.

Actualmente, en la fase inicial de proyecto de un buque se formula un plano de formas a escala 1:50, o 1:100. Este plano se suele obtener por variaciones geométricas de unas formas similares que el astillero haya aplicado ya con éxito en anteriores construcciones, o tomándolo de los datos publicados de modelos ensayados en los canales de experiencias hidrodinámicas, o incluso mediante el uso de series de formas sistemáticas. Sin embargo, si desarrolláramos este plano a escala real, nos encontraríamos con resultados verdaderamente desoladores.

Observaríamos que las líneas tan aparentemente armonizadas, al desarrollarlas quedarían ‘‘abolladas’’ y con faltas de coherencia entre sus diferentes proyecciones del casco.

Por esta razón en cuanto el proyectista ha comprobado que el plano de formas satisface las características del proyecto en relación con el volumen de carena, la posición del centro de carena, y otros parámetros, y además tiene las formas corregidas por el Canal de Experiencias Hidrodinámicas, el primer paso que el proyectista ha de dar es el del *alisado y armonizado* del plano de formas. Esto se puede realizar por alguno de los procedimientos siguientes:

2.1 Alisado en Galibo.

Tradicionalmente este método estaba extendido en los astilleros, especialmente en los de la escuela inglesa. Actualmente este método se puede considerar casi histórico, aunque no debemos dejarlo caer en el olvido.

La oficina técnica preparaba una cartilla de trazado que se enviaba a la *Sala de Gálibos* (Reefing Batten) para que el personal de ésta procediera al alisado y armonizado de formas del buque. Este trabajo se realizaba reproduciendo el plano de formas en la *Sala de Gálibos* a escala real. Como para esto hace falta una superficie plana de grandes dimensiones sobre un suelo de madera, y muchas veces no se disponía de tanto espacio, había que recurrir a dibujar solamente ciertas secciones del casco o bien se podía reducir a ½ o ¼ la escala del longitudinal, obteniendo así una figura deformada que no afectaba a la continuidad de las líneas.

2.1.1 La Sala de Gálibos.

Sus dimensiones debían de ser tales que se pudiera situar sobre el suelo de la sala el plano de formas en verdadera magnitud. Las líneas que se extienden en el sentido de la eslora no era imprescindible desarrollarlas en toda su longitud a escala natural, debido a la existencia del cuerpo cilíndrico. El cuerpo cilíndrico se podía dibujarlo a escala reducida ½ o incluso ¼. La manga y el puntal del buque mayor que se fuera a construir serían los que nos dieran las dimensiones que debería tener nuestra Sala de Gálibos, e insistiendo en que la manga debería ser representada a escala 1:1 o trazarla en dos partes superpuestas.

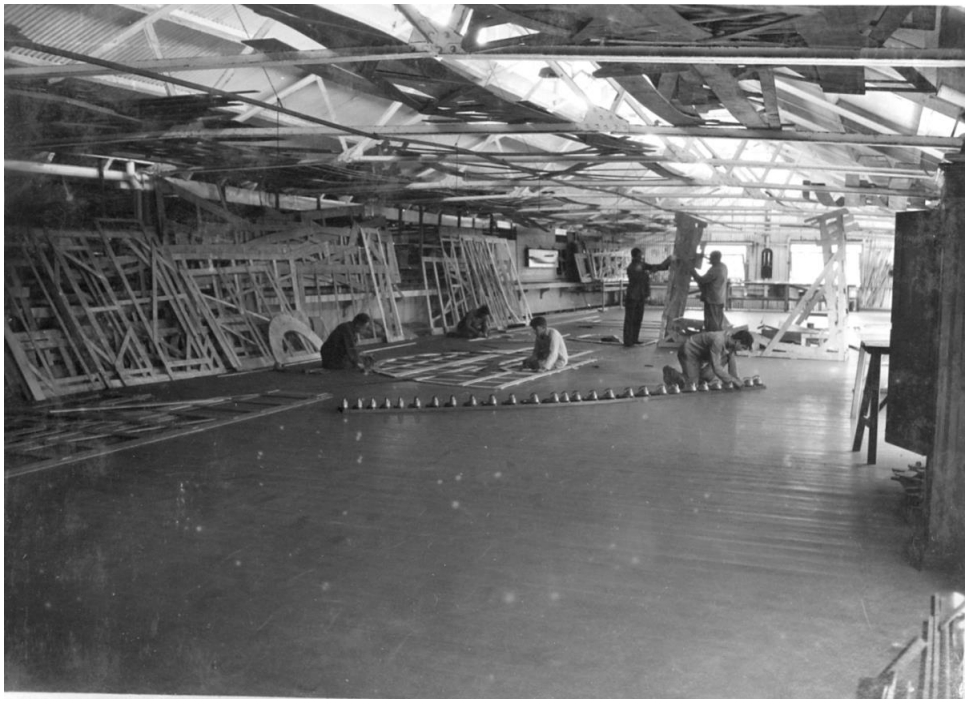


Figura 2.1: Sala de Gálibos.

El espacio disponible en dicha sala, debía de ser tal que se dispusiese de dos zonas en las que trabajar, un lugar donde almacenar las plantillas y aparte de un pasillo que permitiera el paso por la sala estando suficientemente separado de la superficie de trabajo. Con estas premisas la superficie que se necesitaba debía de estar comprendida entre 5000 m^2 a 15.000 m^2, y por supuesto la sala debía de ser cubierta, sin mencionar que a lo largo de ella no debía haber columnas que interrumpieran la zona de trazado.

2.1.2. Medios de trazado.

Para dibujar sobre el piso se empleaba una tiza dura que permitía trazar varias líneas sin que aumentara sensiblemente su espesor.

Las *líneas rectas* se trazaban con un cordón de algodón o lienza impregnada de polvo blanco de tiza que se tensaba pasando por los puntos que la definían y se pulsaba como la cuerda de una guitarra, procurando que esta operación se efectuara en un plano perpendicular al suelo. Para segmentos de poca longitud, de hasta cinco metros, se utilizaban reglas contrastadas. El procedimiento del cordón no puede realizarse sin error apreciable en longitudes grandes, por lo que se marcaban puntos intermedios y la recta se iba trazando en distintos tramos.

Para definir las rectas de grandes longitudes se utilizaban métodos de mayor precisión. El método clásico consistía en tener un fino alambre de acero convenientemente lastrado y apoyado sobre tacos de la suficiente altura en los puntos extremos para que el alambre no rozara el suelo en el punto de flecha máxima de su seno. Para amortiguar las oscilaciones del hilo se suspendían de éste varios pesos colgando de otros hilos y los pesos se sumergían en agua contenida en cubetas. Una vez establecida la línea, con el auxilio de niveles verticales o escuadras se los puntos donde poner los pesos de la recta cada ocho o diez metros. Sirviéndose de estos puntos se trazaba la recta por tramos utilizando la lienza. Este proceso era indispensable para obtener la línea base del trazado y aun así convenía que se comprobara esta línea por métodos más precisos como podía ser el de servirse de un rayo de luz, o de un aparato óptico tal como un nivel o un taquímetro.

Para el trazado de las *curvas* se utilizaban *barrotes, listones o junquillos (splines)*, de sección rectangular o trapezoidal que generalmente eran de madera de abeto sin defectos. En cada curva, la sección del junquillo a utilizar debía ser lo mayor posible para que no sufriera deformación permanente, pero tampoco debía de ser excesiva para que se pudiera manejar cómodamente para adaptarlo a los puntos por donde debía de pasar. Con esto se imponía que las líneas de trazado respondieran a un junquillo deformado elásticamente, criterio fundamental del alisado naval de formas en una Sala de Gálibos

. Para el trazado de líneas de curvatura uniforme, los junquillos podían ser de sección constante. En cambio, si se realizaba un trazado con líneas de curvatura acentuada en los extremos o en el centro, era más práctico utilizar junquillos de sección variable, eso sí, éstos debían ser de mayor rigidez para la parte de las líneas de mayor radio de curvatura.

Los barrotes se colocaban sobre el piso de trazado en el lado cóncavo de la curva y se fijaban por medio de parejas de clavos de cabeza ancha que se podían extraer fácilmente. El barrote debía adaptarse a los puntos de trazado de una forma natural, por lo que se les dejaba cierta libertad. Por ejemplo, supongamos tres puntos sucesivos, suprimiendo toda sujeción de barrote en el punto intermedio la curva debe seguir pasando por él. Si no se consigue de forma natural, una de las operaciones de alisado consistía precisamente en *dejarle libertad al barrote*, a criterio del trazador para que tomara la posición natural que más se aproximara a lo que definen los puntos de la cartilla de trazado. Una vez verificada la línea, se sujetaba mejor al barrote para trazarla.

Por último se insistirá en decir sobre la Sala de Gálibo, que de la precisión con la que se hiciera el trazado, iba a depender luego la exactitud con que se prepare el material. Al tratarse de estructuras de dimensiones tan grandes, las deformaciones y dilataciones tanto térmica como por la humedad del ambiente, han de influir en el trazado y en el puntillaje con que se labraba el material. Debido a esto, cualquier esfuerzo que se haga para mantener estas condiciones lo más uniformemente posible tendrá su compensación pues durante la construcción, especialmente en la de un buque ya soldado, todas las rectificaciones son extraordinariamente costosas. Por supuesto la sala debía de estar bien aislada térmicamente y mejor si se disponía de aire acondicionado.

2.2 Alisado a Escala 1/10.

En los astilleros continentales del centro y norte de Europa, a partir de 1945, se observó una tendencia muy marcada a obtener la cartilla definitiva de trazado sobre un dibujo a escala 1/10.

Los astilleros con práctica en ambos métodos, aseguran que el alisado por este procedimiento es de mayor exactitud que el que se practicaba en la Sala de Gálibos a escala real y el volumen de trabajo se reducía notablemente. Este método se basa en obtener de un dibujo a escala 1/10 una cartilla de trazado. El trazado se realiza en una mesa de grandes dimensiones, uno, dos o tres metros de anchura y una longitud de hasta unos ocho metros.

El trazado de las curvas se hace con junquillos de rigidez adecuada a las curvas que se van a trazar. El personal debe ser muy cualificado, aunque dicha cualificación no es superior al trazador de una Sala de Gálibos.

A partir de la cartilla de trazado preliminar procedente de la sala de proyectos, se dibuja el plano de formas a escala 1/10, con lo cual con este método, al estar el plano de formas escalado, el tiempo a invertir será considerablemente menor. Normalmente se usaban unas 20 secciones de trazado además de las intermedias en las zonas de proa y popa, dándose aparte las dimensiones principales, la separación entre líneas de agua y la clara de cuadernas en cada zona del casco.

El trazado óptico y las máquinas de oxicorte dirigidas electrónicamente con dibujos a escala 1:10 contribuyeron a que éste método se impusiera en su día en todos los astilleros de cierta importancia. A este método se le exige mucha precisión.

2.3 Alisado por métodos numéricos.

Tanto el método de alisado en Gálibos como el método de alisado a escala 1:10 son métodos clásicos para el trazado de las formas, presentando ambos algunas desventajas, las cuales nos han llevado a investigar otros métodos que desarrollaremos a continuación. Entre estas desventajas destacan:

a) Requieren de grandes superficies con equipos costosos y mucha mano de obra.
b) El alisado de líneas depende de los criterios del trazador y no de normas absolutas e independientes del operario.

c) El proceso requiere de varias transmisiones de datos, que inevitablemente, conducirán a errores en el montaje con gastos elevados en las rectificaciones.
d) Necesita mucho tiempo y no se puede esperar a que esté terminado para pedir los materiales, por lo que producirá una necesidad de mayor cantidad de materiales.
e) Es difícil de convertir los datos del trazador en formas adecuadas para controlar las máquinas dirigidas electrónicamente. No es adecuado para el control numérico de las actuales máquinas de corte, y han de someterse a otro proceso intermedio para obtener los valores numéricos.

El alisado por métodos numéricos es también denominado *fairing o faireado* en inglés, o *Lissagre* en francés. Este método se desarrolló gracias al perfeccionamiento y la alta disponibilidad de ordenadores en los astilleros. En este método, la función que representa los valores discretos se desarrolla por aproximaciones de la curvatura, que se define por los puntos que se han dado y que se une por medio de integraciones sucesivas o por el método de mínimos cuadrados, utilizando para ello una función cubica o el ajuste de arcos circulares o parabólicos de manera que el circulo o parábola que representan las curvas pasen por tres puntos consecutivos, siendo el último de los puntos el primero del tramo siguiente.

Los datos que más tarde nos definirán las formas pueden ser introducidos en el ordenador mediante dos caminos diferentes, así podrán ser utilizados para cálculos de arquitectura naval como para ingeniería de producción. Los datos podrán ser introducidos mediante:

- Partiendo de un plano de formas a escala, o de una cartilla de trazado obtenida midiendo sobre dicho plano como mediante series sistemáticas o directamente de un buque o modelo existente.
- Generando directamente las formas a partir de una serie de parámetros fundamentales de la carena, y obteniendo superficies ya lisas definidas mediante ecuaciones que determinan las coordenadas numéricas de cualquier punto de la superficie.

Para obtener la suavidad de las líneas por medio de algoritmos matemáticos, hay que introducir en ellos criterios objetivos. Al realizar el alisado, inevitablemente se modificarán las formas de la carena inicial, y por ello hay que establecer unos criterios en cuanto a los límites permisibles de alejamiento de los valores del plano de formas original de la carena que se ha ensayado previamente en canal, la cual tendrá unas características determinadas.

La definición de la superficie del casco por métodos matemáticos tiene por objeto obtener con rapidez datos precisos que se puedan utilizar en el proyecto del buque y en su construcción, así como en la aplicación al control numérico de las diferentes máquinas y herramientas, como ya se indicó anteriormente.

Podríamos decir que los requisitos exigibles para la correcta definición matemática de las formas con el auxilio del ordenador son:

- Que no introduzcan variación alguna en los parámetros esenciales que caracterizan el proyecto, ni en el comportamiento de la carena.
- La superficie resultante debe satisfacer los criterios de suavidad y alisado.
- Debe lograrse la exactitud y precisión que se exija.
- La superficie del casco se expresa mediante una familia de curvas generadas por ecuaciones matemáticas.
- Su aplicación debe exigir un mínimo esfuerzo manual.

Entre los procedimientos de alisado por métodos numéricos existentes, se pude destacar el AUTOKON como uno de los de mayor difusión e importancia.

2.4 Métodos gráficos iterativos.

Los métodos numéricos que se han citado en el apartado anterior, fueron utilizados, tuvieron su época de ‘‘auge’’, pero sin embargo han sido sustituidos por la utilización de *métodos gráficos interactivos*. Entre éstos podemos destacar como más vanguardistas y con mayor futuro a dos de ellos:

a) *Modelos de alambre*, mediante la definición matemática de líneas que estructuran el buque.
b) *Modelos de superficie*, utilizando la definición de superficies que, mediante “parches”, permiten ‘‘cubrir’’ toda la carena del barco.

A continuación, vamos a analizar cada uno de ellos.

2.4.1 Modelos de alambre.

Representan las formas exteriores del buque como un entramado de líneas. En los modelos de alambre, la definición matemática de las líneas curvas utiliza diferentes procedimientos, entre ellos destacan:

-**La simulación del junquillo**. Utilizan las ecuaciones de la deformada elástica y las líneas obtenidas pasan por todos los puntos que han sido previamente definidos.
Requieren la definición adicional de dos puntos exteriores.

-Bi-arcos. Utilizan las ecuaciones de dos círculos tangentes entre sí, que también pasan por todos los puntos definidos. Adicionalmente a dichos puntos han de definirse dos tangentes en los extremos.

-Splines. Utilizan la ecuación de un polinomio de un cierto grado definido y que pueden pasar o no por todos los puntos definidos. Si pasan o no por los puntos dependerá del tipo de spline a utilizar.

-Curvas de Bezier. Utilizan la ecuación de Bezier, que obtiene líneas tangentes a poligonales de control, cuyos vértices son los puntos definidos, y sólo pasan por los puntos inicial y final.

-B-splines. Utilizan la ecuación de ‘‘B-Splines’’, que obtiene líneas tangentes a poligonales de control cuyos vértices son los puntos de control. Estas curvas se definen por su grado o su orden y por el número de vértices. Con el grado se controlará la proximidad de la tangente a la poligonal. Las curvas tampoco pasan por ninguno de los puntos de control, excepto el inicial y el final. Las curvas de Bezier son un caso particular de este tipo de curvas.

En la siguiente figura se observa el casco de una embarcación definida mediante curvas B-Splines.

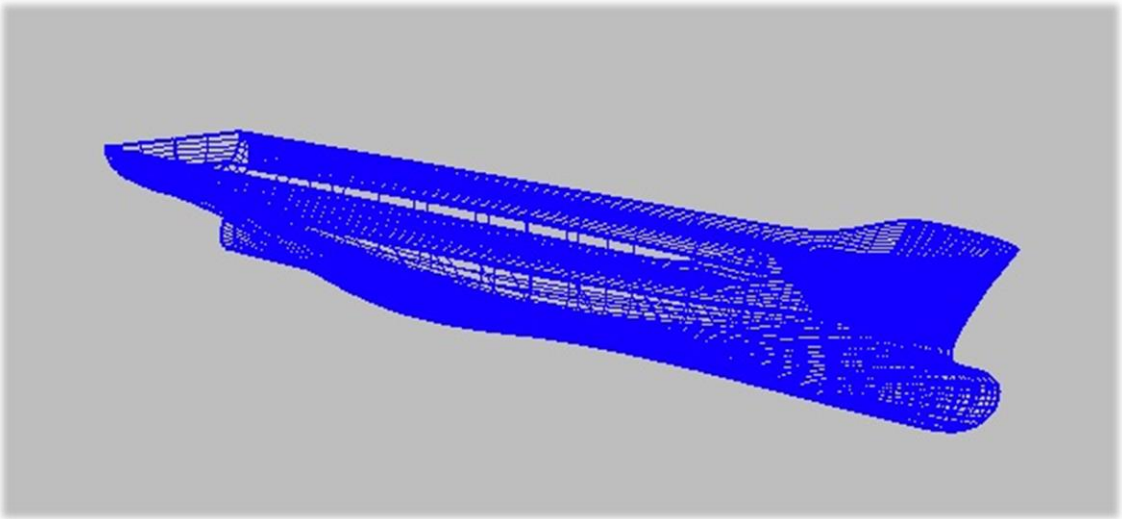


Figura 2.2: Ejemplo de un container definido mediante estructura alámbrica.

En la tabla siguiente se puede apreciar la comparación entre las características más destacables de todos estos procedimientos:

	Junquillo	Bi-arco	Spline	Bezier	B-spline
Dificultad de cálculo	Baja	Baja	Alta	Media	Media
Deformación almacenada	Todos los puntos, dos más	Todos los puntos y dos tangentes	Todos los puntos	Puntos Polígono	Puntos polígono y orden
Calidad del control	Buena	Buena	Pobre	Pobre	Excelente
Tipo de control	Puntos dados	Puntos dados y tangente	Puntos dados y tangentes	Nº y vértices del polígono	Nº y vértices del polígono
Dificultad codillos	Pseudo-curva	Distintas tangentes	Partir curva	Partir curva	Nuevo polígono

Tabla 2.1: Comparación entre los distintos tipos de líneas. Fuente SENER.

Las herramientas de alisado permiten mover, borrar o añadir puntos, según se considere necesario, y el programa realiza el control de los puntos de inflexión que se puedan producir, así como la creación de una malla densa que permita la visualización de las zonas que no tengan un alisado correcto (por ejemplo, mostrando con diferente coloración la zona afectada), fundamentalmente actuando sobre las diferencias producidas en las derivadas primeras.

Estas herramientas también permiten el manejo de diferentes curvaturas mediante el control de las diferencias segundas y, aunque menos utilizado, también el control de las diferencias terceras.

Sin embargo, ningún método es infalible, y éste también genera algunos problemas. El primero que nos encontramos es que al generar un modelo básico, en ocasiones nos encontramos que no resulta éste del todo satisfactorio para su utilización en producción. Cuando hablamos de producción nos referimos a dar las instrucciones precisas a las máquinas de corte y conformado por ejemplo.

Además, este método no define completamente la superficie del casco. Necesitará simularla mediante interpolación de nuevas líneas entre las que ya se han definido. Ésto puede dar lugar a que se definan líneas especiales que no fueran lisas ni precisas. Por último, a día de hoy existen varios caminos para llegar a una solución, es decir, existen diferentes métodos numéricos para obtener las curvas, lo cual puede dar lugar que se obtuvieran coordenadas distintas para un mismo punto.

2.4.2 Modelos de superficie.

En este método se crea y ajusta una *superficie o parche*, la cual debe de pasar por una seria de puntos que definirán los puntos por los cuales deberá pasar el casco del buque. Estos puntos iniciales pueden ser definidos mediante refuerzos estructurales del casco como pueden ser puntales u otros elementos tales como los mamparos, saltillos o la curvatura de la cubierta. Las superficies que utilizan los programas de diseño actuales son:

-Superficies de COONS
.-Superficies de Bezier.
-B-splines.
-Superficies NURBS (Non Uniform Rational B-Splines).

Las principales dificultades de estos sistemas podríamos resumirlas en que su manejo no resulta por el momento demasiado sencillo y requiere un aprendizaje más o menos complejo y prolongado (experiencia). Además, aunque los ‘‘parches” se ajustan muy bien a los puntos que definen esa parte de la superficie, el alisado entre parches resulta también complicado y requiere, como en el caso anterior, de una importante experiencia para resolver los problemas que puedan surgir.

Debido a las dificultades mencionadas anteriormente, se están utilizando modelos mixtos, en los que se manejan líneas para introducir la información y para realizar las modificaciones, posteriormente a partir de estas líneas ya definidas se crea un superficie que cubrirá a las mismas como vemos en la siguiente figura.

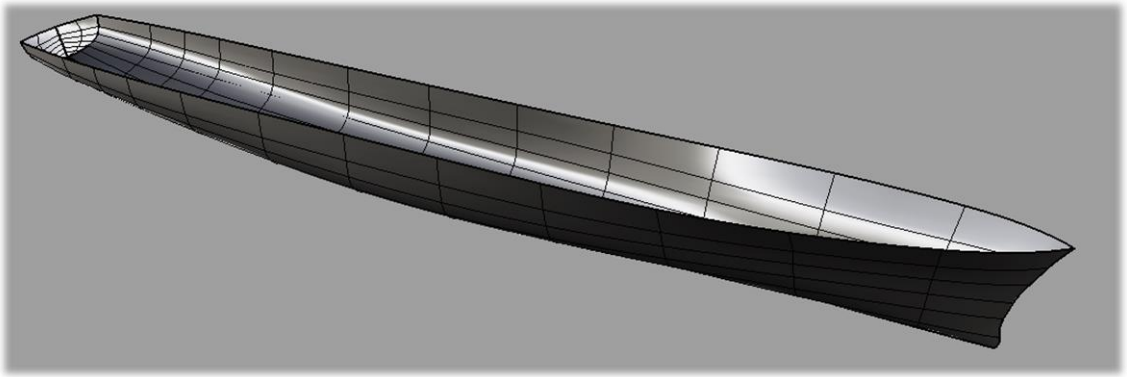


Figura 2.3: Ejemplo de una corbeta definida mediante un modelo de superficie. Maxsurf.

2.5 Métodos de comprobación del alisado.

A la hora de asegurarse de qué formas de una embarcación están perfectamente lisas, no se puede acudir a herramientas como el renderizado. Cuando un usuario mira un casco renderizado y coloreado en la pantalla no puede apreciar lo que realmente se esconde detrás, en la mayoría de los casos un casco que presenta abolladuras.

Una imagen renderizada puede parecer a simple vista lisa, esto es debido a que los programas de diseño procesan las imágenes descomponiéndolas en triángulos y las rutinas de sombreado suavizan los bordes de los triángulos, sin embargo la geometría que realmente presenta una imagen renderizada y de apariencia lisa puede no ser útil para la construcción del buque.

Los programas de diseño CAD, como Rhinoceros, incorporan ciertas herramientas con las cuales es fácil comprobar la calidad del alisado realizado por el usuario, entre ellas encontramos las siguientes:

La *comprobación visual* mediante la aplicación de un rayo de luz sobre el casco renderizado o bien mediante un renderizado de tipo *entorno*, donde se aplica una textura metálica donde es fácil evaluar la existencia de *bollos* como vemos en la figura 2.4:

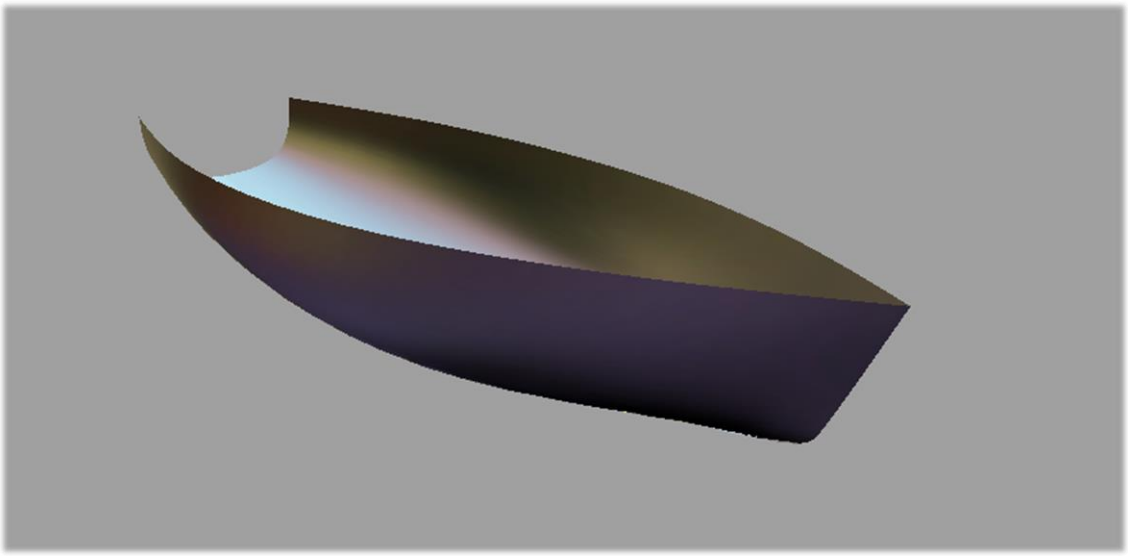


Figura 2.4: Comprobación del alisado mediante entorno, plata a pinceladas.

También se puede comprobar el alisado mediante una *superficie del tipo cebra*, como vemos en la figura 2.5. Constituye otra opción para comprobar el alisado del modelo. Las regiones con una intensidad de luz reflejada constante se sombrean en bandas. Esto es similar a la manera en que el ojo humano detecta puntos discordantes en una superficie, puesto que el brillo y las sombras varían en esas áreas. Si los bordes de las rayas de la cebra se curvan suavemente entonces la superficie es lisa en estas regiones. En las líneas de codillos varían abruptamente.

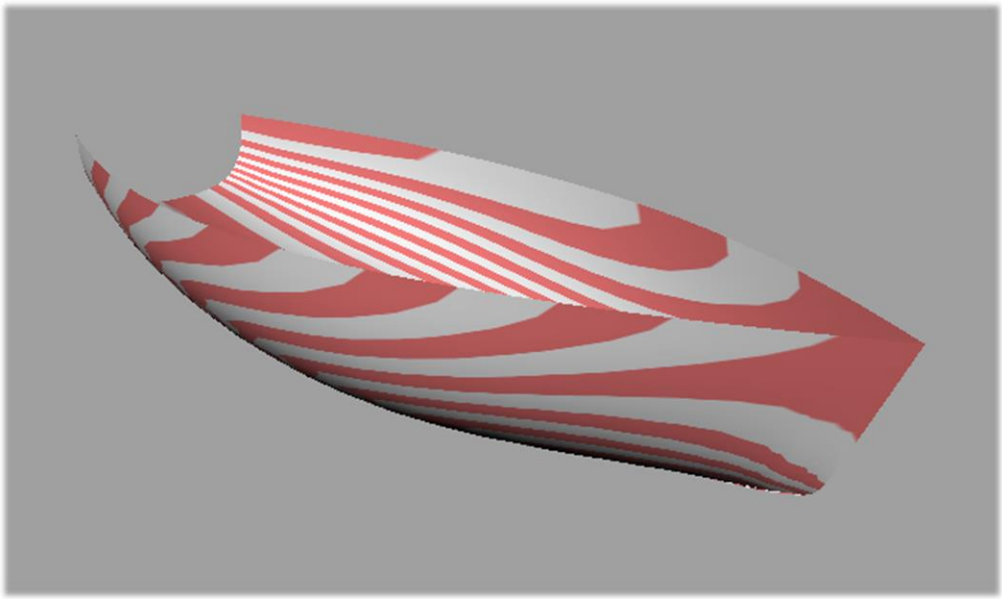


Figura 2.5: Comprobación del alisado mediante una superficie cebra.

Otro medio de comprobación, es a través de un análisis de la *curvatura*. El modelo se sombrea en colores, basados en la Curvatura Gaussiana Discreta en cada punto. Los cascos están curvados en dos direcciones, denominadas curvaturas principales. La Curvatura Gaussiana es el producto de esas dos curvaturas principales.

Por último, el método más eficiente es comprobando la curvatura mediante curvas de control. Estas curvas pueden ser definidas como e*l índice de cambio (en un punto) del ángulo entre una curva y la tangente a la curva.*

Si el trazado de curvaturas se usa e interpreta correctamente es posible producir un alisado perfecto de la superficie.

En la siguiente figura se muestra un ejemplo de curva de control de un yate a vela. La parte superior de la imagen muestra un alisado deficiente de la curva. En ella se puede ver un cambio en el signo de la curvatura muy pronunciado, seguida de un súbito incremento en la medida de la curvatura. Después de esto la medida de la curvatura disminuye rápidamente hasta que se vuelve a incrementar hacia la proa. La mitad inferior de la imagen muestra la misma curva de control después de alisarse bien. Es obvio que la curvatura cambia gradualmente ahora y que la curva es muy suave.

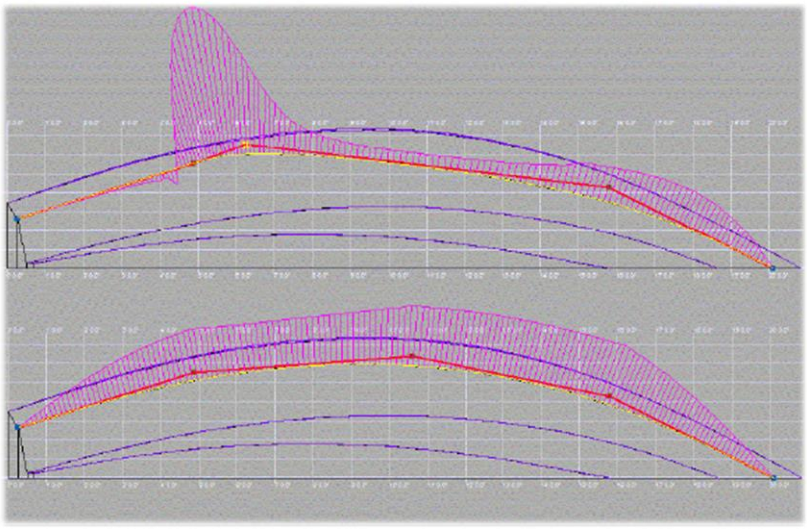


Figura 2.6: Control de la curvatura mediante curvas de control.

Como ya se ha podido comprobar, a la hora de diseñar cualquier embarcación mediante la ayuda del ordenador, éste va a necesitar como base disponer de un algoritmo de cálculo para las líneas curvas y además se recomienda que el diseñador conozca perfectamente cómo funcionan las curvas que componen la superficie que pretende alisar. En este proyecto hemos decidido centrarnos en los splines, los cuales serán descritos con mayor grado de detalle en los siguientes capítulos.

Capítulo III

Splines

Capítulo III

Splines

Como ya se ha descrito en capítulos anteriores, las curvas con las que se trabaja en el diseño naval son bastante complejas, y en la mayoría de los casos no podrán ser expresadas analíticamente mediante una única función. Si nos planteáramos el expresar esas curvas mediante una única expresión analítica, nos daríamos cuenta que eso plantea muchos problemas. Por un lado, la forma de la función polinómica de un grado alto, no suele adaptarse de la manera deseada debido al gran número de extremos e inflexiones. Además, al intentar buscar esta expresión, nos daremos cuenta de que su cálculo es bastante complejo, lo que limita su utilidad en análisis numérico.

Por todas estas razones, es más conveniente dividir el intervalo de interés en subintervalos más pequeños y a la vez usar en estos subintervalos polinomios de un grado relativamente bajo, tratando de que la función a trozos definida de este modo tenga un aspecto final adecuado a las curvas que estamos buscando. De este modo cada fragmento será suave, pero tendremos que prestar mucha atención para que mantengan esa suavidad en las zonas donde se empalman los fragmentos.

Llegados a este punto dentro del subcampo matemático del análisis numérico definiremos el concepto ‘’Spline’’ como una curva diferenciable definida en porciones o trozos mediante polinomios. Los splines más utilizados son los de grado tres o cúbicos, ya que éstos ofrecen una relación buena entre flexibilidad y velocidad de cálculo; comparando con polinomios de orden superior. Los splines requieren de menos cálculos y menos memoria, a la vez que son más estables.

3.1 Definición de función la función Spline.

Supongamos que en un segmento [a,b] está dado un retículo

$$w: \ a = x_1 < x_2 < \cdots < x_{m-1} < x_m = b.$$

Los puntos x_1 y x_m se llaman *nodos fronterizos* del retículo w, y los puntos $x_2, \dots, x_{m-1}$ son *nodos interiores*.

Una función $S(x)$, definida en el segmento $[a, b]$, se llama *Spline* de orden $p+1$ (de grado p) si:

1) En todo segmento

$$\Delta_i = [x_i, x_{i+1}] \ \forall \ i = 1, \dots, m-1$$

es un polinomio de grado$\leq \ p$:

$$S(x) = S_i(x) = \sum_{k=0}^{p} a_k^{(i)} (x - t_i)^k \quad \forall \ i = 1, \dots, m-1.$$

En todo el segmento, el Spline es un polinomio de grado p con $p+1$ coeficientes. En total se tienen $(m-1)$ segmentos parciales. Consecuentemente, para determinar completamente el Spline es necesario hallar $(m-1)(p+1)$ números.

2) Es $(p-1)$ veces diferenciable con continuidad en el segmento $[a, b]$:

$$S(x) \in \ C^{p-1}[a, b].$$

Esta condición implica la continuidad de la función $S(x)$ y sus derivadas $S'(x), S''(x), \dots, S^{p-1}(x)$ en todos los $m-2$ nodos interiores del retículo w. Por tanto, para hallar los coeficientes de todos los polinomios se dispone de $p(m-2)$ condiciones (ecuaciones).

Para determinar completamente el Spline faltan $(p+1)(m-1)-p(m-2)=(m+p-1)$ condiciones.

La elección de las condiciones adicionales dependerá del carácter del problema analizado.

3.2 Splines frente a otros métodos de interpolación.

En el ámbito de la ingeniería, estamos bastante familiarizados con los métodos numéricos, conociendo algunos otros métodos de interpolación que pueden ser útiles para resolver ciertos problemas. Llegado a este punto, vamos a explicar el porqué de la elección de los splines a través de un sencillo ejemplo.

Un método numérico bastante conocido es la *interpolación polinómica de Lagrange*. La forma general del polinomio será:

$$P(x)=\sum_{i=1}^{n} y_i \prod_{\substack{j=1 \\ j\neq 1}}^{n} \frac{x-x_j}{x_i-x_j}$$

Ahora bien, vamos a realizar una comparación entre el polinomio de Lagrange y los splines cúbicos. Supongamos la función:

$$y=\frac{10}{\mathrm{x}^2+\mathrm{x}+1}$$

Representando dicha función en un determinado dominio, como se puede ver en la figura 3.1:

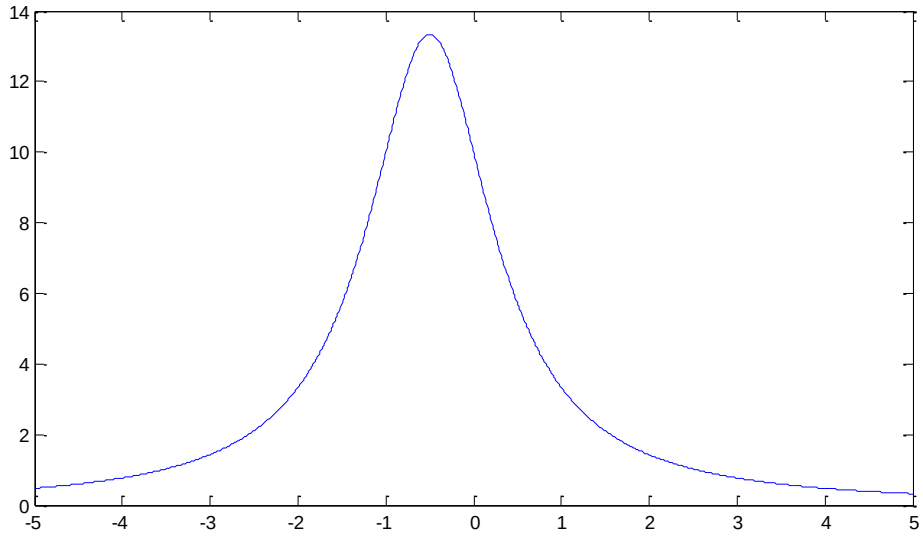


Figura 3.1 : Gráfica de la función obtenida mediante Matlab

A continuación realizamos la interpolación polinómica de Lagrange para 5, 7 y 15 puntos y superponemos ambas gráficas.

Para dichos cálculos se ha usado el programa Maple, el cual es un programa matemático de propósito general capaz de realizar cálculos simbólicos, algebraicos y de álgebra computacional. Fue desarrollado originalmente en 1981 por el Grupo de Cálculo Simbólico en la Universidad de Waterloo en Waterloo, Ontario, Canadá.

Para 5 puntos, (figura 3.2):

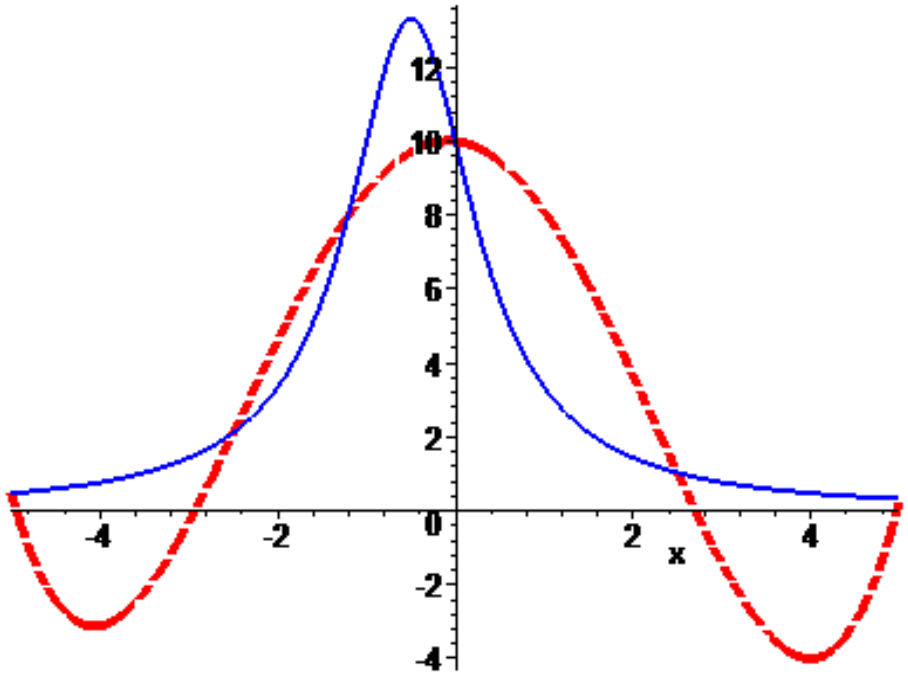


Figura 3.2: Superposición entre la función original (azul) y la interpolación de Lagrange para 5 puntos. Maple

Para 10 puntos (figura 3.3):

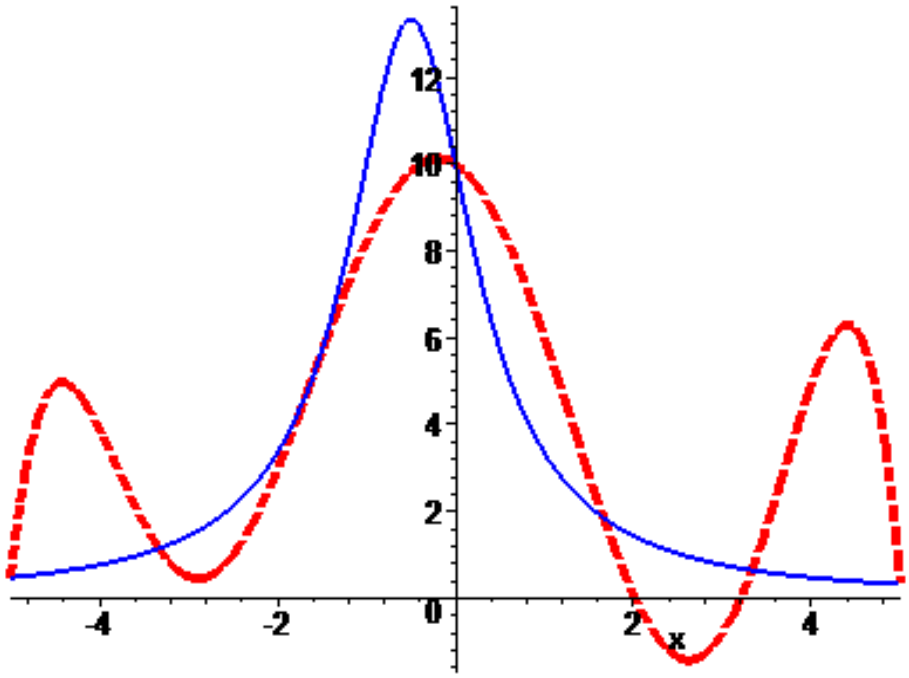


Figura 3.3: Superposición entre la función original (azul) y la interpolación de Lagrange para 10 puntos. Maple

Para 15 puntos, (figura 3.4):

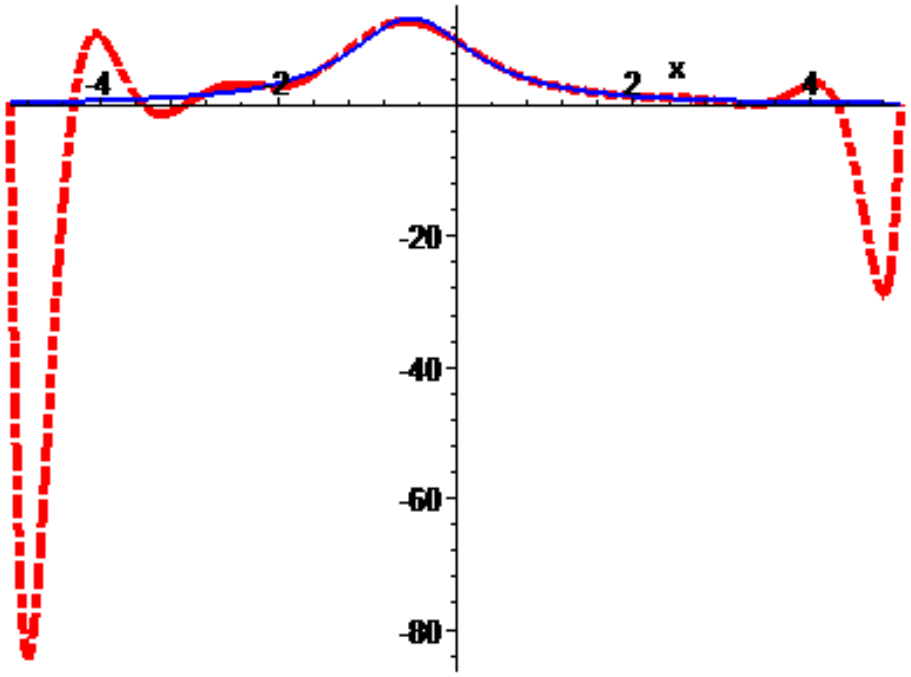


Figura 3.4: Superposición entre la función original (azul) y la interpolación de Lagrange para 15 puntos. Maple.

Se observa como en el primer caso, figura 3.2, el polinomio que se obtiene es de tercer grado, y la aproximación es bastante mala, el polinomio de Lagrange se limita a pasar por los puntos. Con 7 puntos, figura 3.3, la aproximación entre el intervalo -2 y 1 es buena, y finalmente con 15 puntos, figura 3.4, donde la aproximación en el centro es muy buena, pero se aprecia la aparición de unas ondulaciones en los extremos lejos de la propia función. Este fenómeno es conocido como ‘’fenómeno Runge’’.

No obstante, realizando una *interpolación segmentaria de Lagrange*, se podría conseguir que la función y la aproximación fueran coincidentes, aunque no se conseguiría continuidad de la primera derivada en los extremos de los segmentos, sin embargo esta condición de continuidad quedará satisfecha mediante el uso de los splines.

Al realizar la interpolación mediante splines, usando la función propia spline en Maple, se observa que con 15 puntos la interpolación es prácticamente perfecta y casi coincidente con la función original, véase figura 3.5.

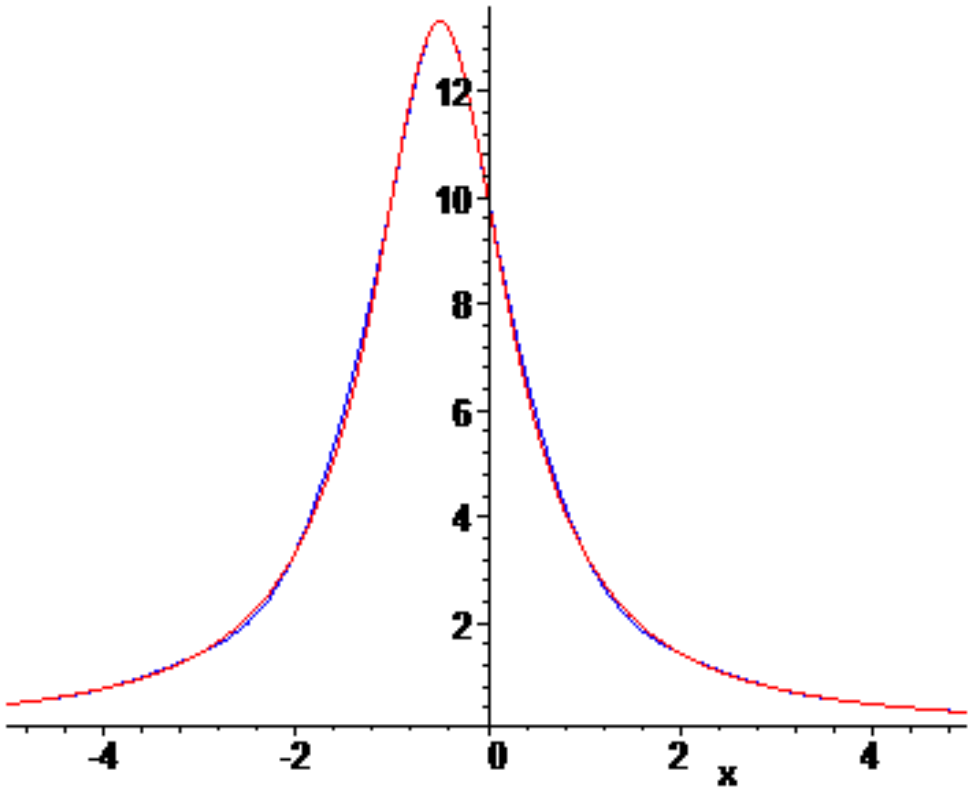


Figura 3.5: Superposición entre la función (azul) y los splines (rojo). Maple.

Como conclusión se obtiene que:

- Hay que realizar muchas operaciones aritméticas en otros métodos de interpolación (como el de Lagrange), ya que habrá 2 bucles entrelazados, uno para el sumatorio y otro para el productorio.
- Si queremos añadir o suprimir un punto al conjunto de datos, habrá que volver a hacer todos los cálculos (este problema también lo presentan las splines naturales, aunque otras splines no lo tienen).
- En ciertos casos, un polinomio de grado alto puede desviar mucho la curva que pasa por los puntos dados (fenómeno de Runge).

3.3 Aplicaciones de los Splines.

Proporcionaremos un conjunto de puntos conocidos, que serán una serie de coordenadas, los cuales denominaremos puntos de control. Estos puntos de control son los que servirán de directrices y con los que después ajustaremos funciones polinómicas continuas mediante una de las siguientes formas que se pueden ver representadas en la figura 3.7. y 3.8.

a) La curva realiza **interpolación** del conjunto de puntos de control cuando las secciones polinómicas se ajustan de modo que la curva pasa a través de cada punto de control $(x_i, y_i)\ \forall\ i = 1, \dots, m$; lo que proporciona m condiciones. Las restantes $p - 1$ ecuaciones para la construcción del spline mediante interpolación serán los valores de las derivadas menores del Spline en los extremos del segmento analizado: condiciones de contorno o de frontera.

b) La curva realiza una **aproximación o suavizamiento** al conjunto de puntos de control cuando los polinomios se ajustan a la trayectoria general del punto de control *sin pasar necesariamente a través de ningún punto de control* $(x_i, y_i)\ \forall\ i = 1, \dots, m$. La medida de esta cercanía se puede definir de diferentes maneras, lo que genera una gran variedad de *splines suavizantes.*

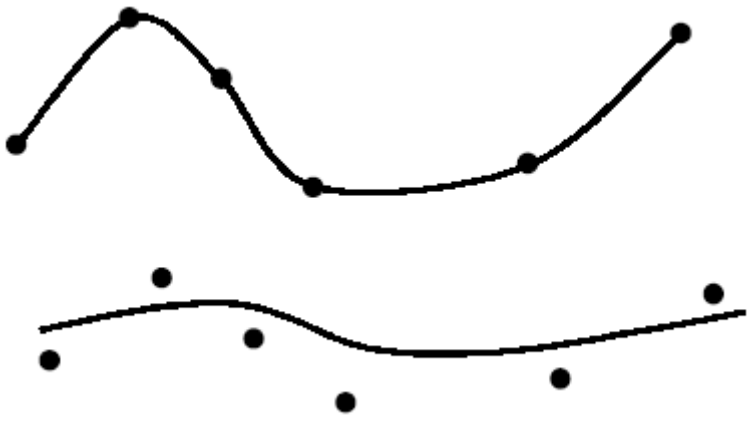


Figura 3.6: Representación de una curva que realiza una interpolación (superior) y otra que realiza un suavizamiento (inferior).

Una curva spline se define y se modifica con operaciones sobre sus puntos de control. Los programas de diseño CAD, además podrán insertar puntos de control adicionales para ayudar al diseñador en el modelado y alisado.

El problema de *suavizamiento* que vamos a plantear consistirá en la obtención de una función suave a partir de sus valores en ciertos puntos. Está claro que este problema tiene infinitas soluciones diferentes. Imponiendo sobre la función condiciones adicionales, se podrá lograr la univocidad.

En los siguientes capítulos nos centraremos en los problemas de suavizamiento mediante splines cúbicos suavizantes o los denominados ''smoothing splines''.

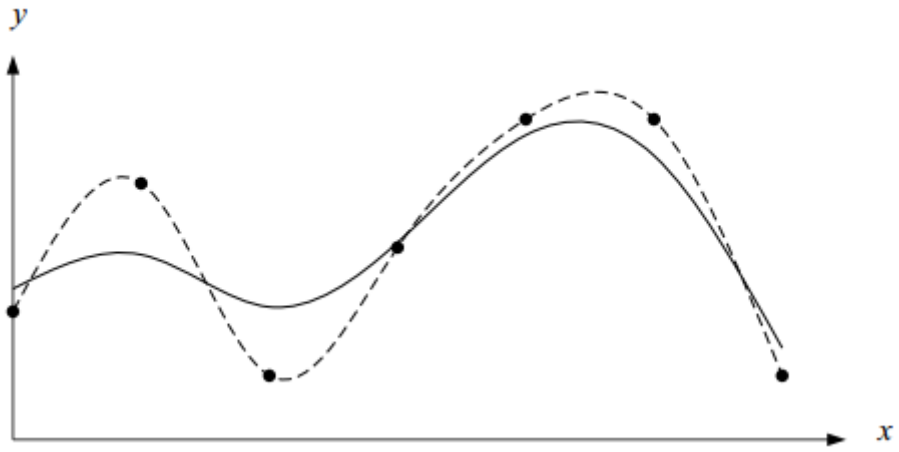


Figura 3.7: Splines interpolantes (discontinuo) y Splines Suavizantes (continuo).

3.4 Splines Suavizantes y la analogía del muelle en el dibujo naval.

Hasta ahora se ha demostrado que las curvas suaves son necesarias en el dibujo naval, sin embargo en la construcción naval los ‘’splines’’ han sido utilizados desde siglos atrás mediante el uso de los denominados ‘’junquillos’’.

Figura 3.8: Junquillo y pesos.

El junquillo (figura 3.8), que ya se ha descrito en el proceso de delineación en la Sala de Gálibos, como sabemos, es un listón flexible con el cual se pueden dibujar curvas suaves en dos dimensiones. Mediante el ajuste de los puntos finales del spline o junquillo y aplicando unos *pesos*, se creará una curva suave, sobre la cual se puede realizar el trazo. La curva creada mediante el junquillo dependerá básicamente de dos factores:

-La *rigidez* del material con el que está hecho el junquillo.
-De los *pesos* y los puntos donde estos están aplicados.

Las superficies que compondrán el casco de una embarcación, se definirán por la posición de un conjunto de puntos de control que de forma global constituyen la malla de puntos de control. *El movimiento de estos puntos de control permite manipular una superficie hasta obtener la forma deseada*.

Los puntos de control que servirán como datos de partida, normalmente podrán ser obtenidos de diversas maneras. En el caso de ser obtenidos directamente de la cartilla de trazado, las coordenadas de los nodos serán exactas, en este caso es útil realizar una interpolación.

En el caso de obtener unas mediciones de los nodos de forma manual, como puede ser midiendo directamente en una embarcación o realizando el proceso de escanear un plano de formas para luego trabajar con él, es decir asumiendo cierto ‘’ruido’’ en las mediciones, es mucho más útil realizar un proceso de suavizamiento.

La base del proceso de modelado de diseños mediante la ayuda del ordenador está en comprender cómo se pueden usar los puntos de control para obtener superficies con la forma deseada. Esto se explica de forma más clara con la siguiente analogía.

A continuación vamos a describir dos procesos básicos que se pueden realizar con el junquillo, una interpolación y un suavizamiento:

En un problema de *interpolación*, como se observa en la figura 3.9, el junquillo se colocará recto en la mesa donde se realizan los trazados.

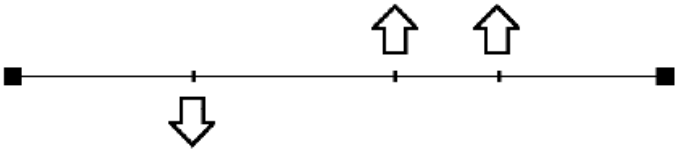


Figura 3.9: Spline ajustado mediante pesos. Junquillo.

A continuación se arrastra el spline en un número finito de puntos mediante la ayuda de unos pesos, la rigidez inherente del material con el que está confeccionado el spline dará como resultado una curva suave, la cual es usada para dibujar (figura 3.10).

Figura 3.10: Spline ajustado mediante pesos. Junquillo.

Estos puntos son los denominados puntos de edición, que son como los puntos de control excepto que siempre están situados en la curva y mover un punto de edición generalmente cambia la forma de toda la curva (mover un punto de control sólo cambia la forma de la curva en una subregión). Los puntos de edición son más útiles cuando se necesita un punto en el interior de una curva para que atraviese exactamente una determinada posición.

La edición de puntos de control es preferible cuando se quiere cambiar la apariencia de una curva y mantener la "lisura" es importante.

Para generar estas curvas, en este proyecto fin de carrera utilizaremos una ecuación matemática, el *Spline Suavizante*, que simulará lo que ocurre con los junquillos flexibles. Las curvas spline se definen por la posición de sus puntos extremos, la posición y número de puntos de control a lo largo de la curva y la flexibilidad del junquillo. Para definir una curva spline será necesario al menos definir dos extremos y un punto de control.

En un problema de *suavizamiento* como en el que nos vamos a centrar, en lugar de hacer pasar la curva spline a lo largo de unos puntos, nos centraremos en el efecto de atraer la curva spline mediante unos resortes, véase figura 3.11.

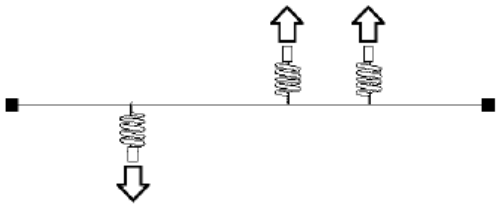


Figura 3.11: Spline ajustado mediante resortes.

Al mover un punto de control, la rigidez inherente de los splines y los resortes o muelles es combinada para mantener la curva suave. Este efecto se puede observar ya que lo curva no pasará necesariamente por cada punto de control, si no que será atraída en mayor o menor medida hacia los puntos de control. La curva resultante es suave con sólo los dos puntos finales de control fijos situados en la curva, figura 3.12.

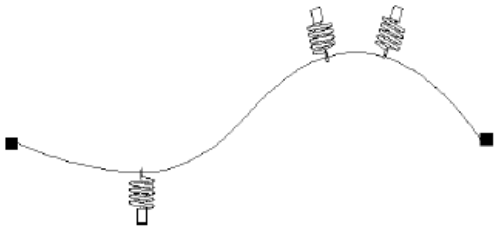


Figura 3.12: Spline ajustado mediante resortes o muelles.

Moviendo los puntos de control a una u otra posición, es posible curvar el junquillo hasta una forma dada. La curvatura del junquillo estará libre de irregularidades debido a la elasticidad de los muelles y la flexibilidad del propio junquillo. Si el junquillo se hiciera más rígido o flexible, la curvatura sufriría un cambio en su magnitud.

Como hemos expuesto, los puntos de control no se hallan sobre la curva creada, sino que la curva es atraída hacia la posición de los puntos de control, y entonces los puntos por los que ha de pasar la curva sólo coinciden con los puntos de control en los extremos.

Los puntos de control tienen una influencia local sobre la curva, de modo que el desplazamiento de cualquiera de ellos modifica sólo la zona de la curva sobre la que tiene influencia, y no la curva completa. Están dispuestos en forma de malla rectangular con filas y columnas, de modo que cada uno pertenece a una fila y a una columna de la malla. Debido a esto en el dibujo asistido por ordenador podemos encontrar dificultades a la hora de realizar el alisado, ya que no es fácil entender como variará la curva al mover uno de estos puntos, por ello el alisado naval es una operación manual que requiere cierto grado de experiencia.

Aunque éste es un ejemplo en dos dimensiones, los programas de dibujo asistido por ordenador usan un procedimiento análogo para generar sus splines en tres dimensiones para formar superficies, véase figura 3.13.

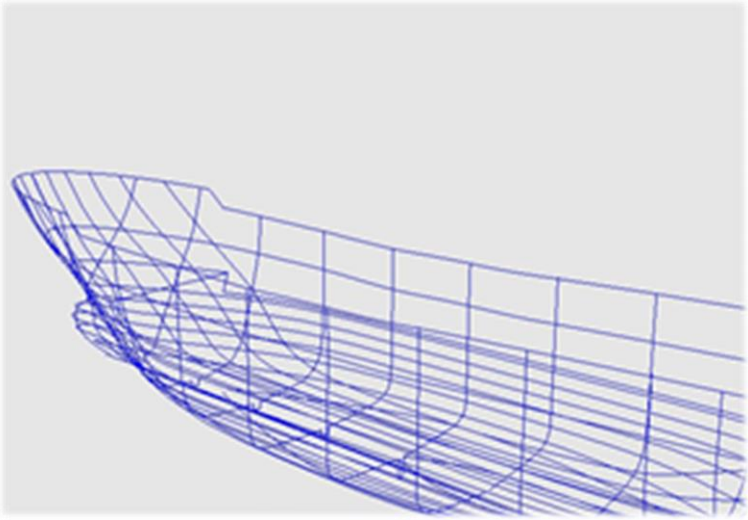


Figura 3.13: Detalle de splines en 3D.

Del mismo modo que una fila de puntos de control bidimensionales puede definir una curva bidimensional, una red de puntos de control tridimensionales puede definir una superficie tridimensional completa. Si se considera una malla tridimensional de puntos de control es posible imaginar los junquillos situados a lo largo y a lo ancho de la red, definiendo de este modo una superficie. Así una superficie se creará generando los junquillos en tres dimensiones a partir de los puntos de control que constituyen la malla.

Por último, es preciso recordar que, para producir un cambio en la superficie, hay que cambiar la posición relativa de los puntos de control de la malla. El programa que se utilice para realizar el diseño recalculará y mostrará entonces la nueva forma de la superficie. Ocurre del mismo modo que con la analogía del muelle: sólo moviendo los puntos de control es posible cambiar la forma de la superficie, y no moviendo directamente los puntos de la propia superficie.

Capítulo IV

Splines Cúbicos Suavizantes

Capítulo IV

Splines Cúbicos Suavizantes

4.1 Planteamiento del problema de suavización.

Consideramos el retículo

$$a = x_1 < x_2 < \cdots < x_{m-1} < x_m = b,$$

y la colección de números

$$y_1, y_2, \dots, y_{m-1}, y_m.$$

Puede suceder que los valores y_i en el arreglo

$$(x_i, y_i), \quad \forall \quad i = 1, \dots, m,$$

contengan cierto error. Esto significa que para todo $i = 0,1, \dots, m$ existe un intervalo

$$(c_I, d_I) \text{ ó } bien \quad (y_{I-\delta}, y_{I+\delta}),$$

tal que cualquier número perteneciente a él puede ser tomado como valor de y_i. Los valores y_i pueden ser, por ejemplo, los resultados (que contienen un error aleatorio) de las mediciones de cierta función $y(x)$ para los valores dados de la variable x. No es conveniente utilizar interpolación para construir la función $y(x)$ a partir de estos valores ''experimentales'', por cuanto la función interpolante reproducirá las oscilaciones condicionadas por la componente aleatoria en el arreglo $\{z_i\}$.

En este caso es más apropiado el método de suavización, uno de cuyos objetivos es disminuir la aleatoriedad en el resultado de las mediciones. Habitualmente, en tales problemas se pide hallar una función cuyos valores para $x = x_i, i = 1,2, \dots, m,$ pertenezcan a los intervalos correspondientes y que tenga, además propiedades bastante buenas (por ejemplo, derivadas primera y segunda continuas, su gráfico no es muy encorvado, es decir no tiene oscilaciones muy fuertes, etcétera).

Un problema similar tiene lugar también al construir, a partir de un arreglo (exacto)

$$(x_i, y_i), \quad \forall \quad i = 1, \dots, m,$$

una función que pase no por los puntos dados, sino cerca de ellos y que, además, varíe de manera suficientemente suave. En otras palabras, es como si la función buscada suavizara el arreglo sin interpolarlo.

4.2 Spline Cúbico Suavizante. Definición.

Consideremos un retículo

$$\omega \ : \ a = x_1 < x_2 < \cdots < x_{m-1} < x_m = b,$$

y dos colecciones de números

$$y_1, y_2, \dots, y_{m-1}, y_m.$$

Se denomina *spline cúbico suavizante* en un retículo ω a una función S(x) que:

1) En todo el intervalo

$$[x_i, x_{i+1}], \qquad \forall \quad i = 1, \dots, m-1,$$

es un polinomio de tercer grado

$$S(x) = S_i(x) = a_0^i + \ a_1^i \ (x - x_i) + \ a_2^i \ (x - x_i)^2 + \ a_3^i \ (x - x_i)^3 \ . \qquad (4.1)$$

En todo el intervalo, el spline es un polinomio de tercer grado que se define mediante cuatro coeficientes. En total hay $(m-1)$ intervalos, por lo que, para definir completamente el Spline es necesario hallar $4(m-1)$ números:

$$a_0^{(i)}, a_1^{(i)}, a_2^{(i)}, a_3^{(i)}, \quad \forall\, i = 1, \dots, m-1.$$

2) Es dos veces diferenciable con continuidad en el intervalo [a,b], es decir, pertenece a la clase C^2[a,b].

Esta condición significa continuidad de la función $S(x)$ y de sus derivadas $S'(x)$ y $S''(x)$ en todos los nodos internos del retículo w. Como el número de nodos internos es $m-2$, entonces tenemos $3(m-2)$ condiciones.

3) En ella alcanza su *mínimo el funcional*

$$J(f) = \int_a^b (f''(x))^2 dx \; + \; \sum_{i=0}^{m} \frac{1}{\rho_i} (f(x_i) - y_i)^2, \tag{4.2}$$

donde y_i y $\rho_i > 0$ son números dados. A los números ρ_i se les denomina pesos.

4) Satisface *condiciones de contorno* de uno de los tres tipos siguientes, explicadas a continuación.

4.3 Condiciones de contorno.

Las condiciones de contorno se dan en forma de restricciones sobre los valores del spline y de sus derivadas en todos los nodos fronterizos del retículo ω.

A. Condiciones de contorno de primer tipo.

Las derivadas primeras de $S(x)$ son conocidas en los extremos del intervalo [*a,b*]:

$$S'(a) = z_1' \quad S'(b) = z_m'. \tag{4.3}$$

B. Condiciones de contorno de segundo tipo.

Las derivadas segundas de $S(x)$ son conocidas en los extremos del intervalo [*a,b*]:

$$S''(a) = z_1'' \quad S''(b) = z_2''. \tag{4.4}$$

C. Condiciones de contorno de tercer tipo.

$$S(a) = S(b),\ \ S'(a) = S'(b),\ S''(a) = S''(b). \tag{4.5}$$

Estas condiciones se denominan *periódicas.*

Teorema: El Spline cúbico $S(\mathrm{x})$ que minimiza el funcional y satisface las condiciones de contorno de uno de los tres tipos indicados está definido unívocamente.

Por último el spline cúbico que minimiza el funcional *J(f)* y satisfice las condiciones de contorno de *i-ésimo* tipo se denomina *spline suavizador de i-ésimo tipo.*

4.4 Elección de las condiciones de contorno.

La elección de las condiciones de contorno es uno de los problemas principales en los problemas de interpolación y aproximación mediante splines, y adquiere una importancia especial cuando es necesario garantizar una precisión alta del spline $S(x)$ en las proximidades de los extremos del intervalo $[a, b]$.

El efecto sobre la función spline en función de las condiciones de contorno elegidas, disminuirá conforme estemos más cerca de los puntos intermedios a interpolar, sin embargo nuestra función dependerá en los extremos en gran medida de las condiciones elegidas. La elección de las condiciones de contorno depende, a menudo, de la existencia de datos adicionales sobre el comportamiento de la función spline.

Si se conocen los valores de la derivada primera $f'(x)$ en los extremos del intervalo $[a, b]$, es decir, se conoce la dirección de la tangente de la curva en los extremos; entonces es lógico utilizar condiciones de contorno de primer tipo.

Si por el contrario, lo que se conocen son los valores de la derivada segunda $f''(x)$; entonces es lógico utilizar condiciones de contorno de segundo tipo.

Si existe la posibilidad de elegir entre condiciones de primer y segundo tipo, se dará preferencia a las primeras.

Si la función es periódica, se deben elegir condiciones de contorno de tercer tipo.

En caso de no existir información adicional sobre el comportamiento de la función, se pueden utilizar las condiciones naturales de contorno:

$$S''(a) = 0 \quad ; \quad S''(b) = 0.$$

Con estas condiciones la precisión en la aproximación puede disminuir bruscamente cerca de los extremos del intervalo $[a, b]$.

Otra opción es utilizar las condiciones de contorno de primer o de segundo tipo con valores aproximados; esto quiere decir, que se tomarán sus aproximaciones en diferencias, en vez de los valores exactos de sus derivadas.

4.5 Elección de los coeficientes de peso.

La elección de los coeficientes de peso ρ_i del funcional permite controlar, en cierta medida, las propiedades de los splines suavizantes.

Para ilustrar cómo influye este parámetro de peso o también llamado de suavizado, consideraremos una serie de puntos pertenecientes a un retículo, los cuales serán ajustados mediante 6 valores distintos de ρ.

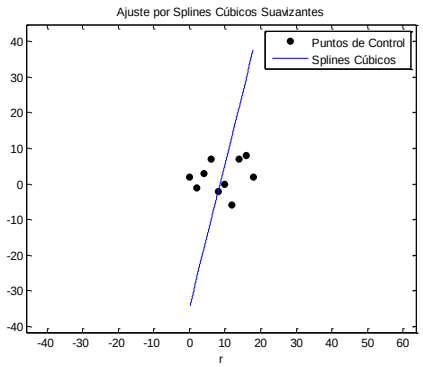


Figura 4.1: Spline cúbico suavizante para $\rho = \infty$

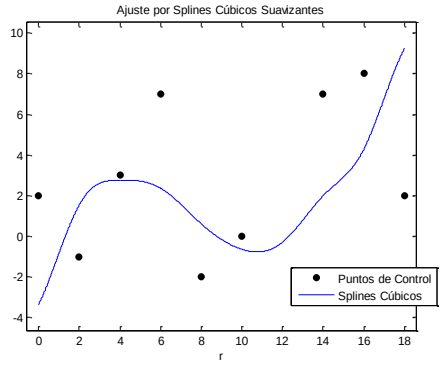


Figura 4.2 Spline cúbico suavizante para $\rho = 10$

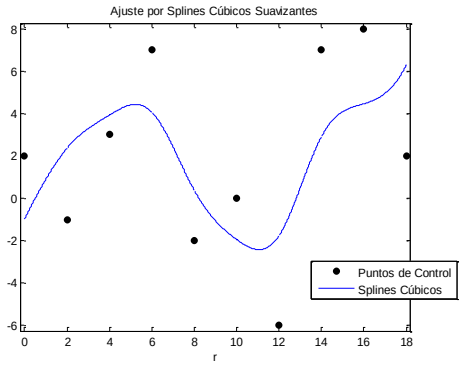


Figura 4.3: Spline cúbico suavizante para ρ = 1.5

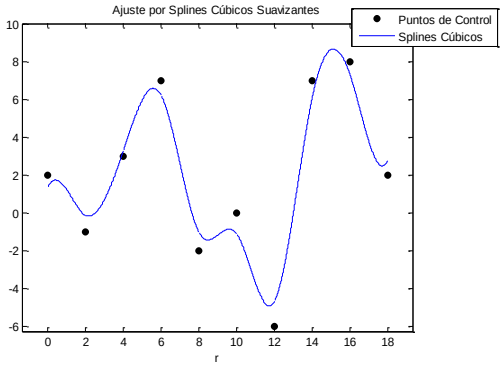


Figura 4.4: Spline cúbico suavizante para ρ = 0.1

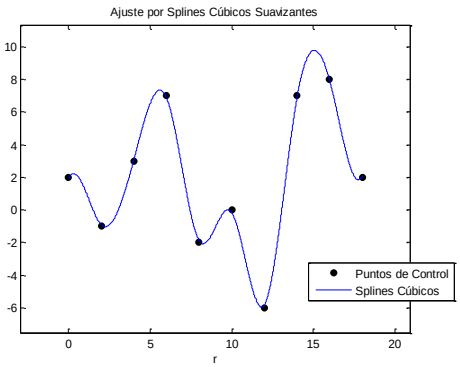


Figura 4.5: Spline cúbico suavizante para ρ = 0.01

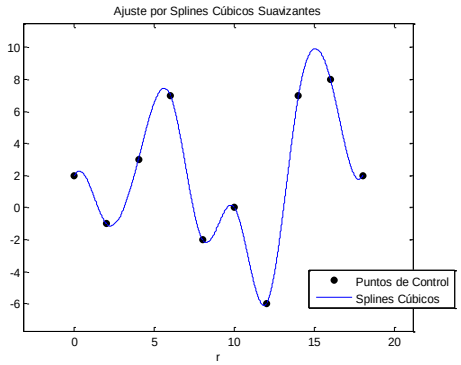


Figura 4.6: Spline cúbico suavizante para ρ = 0

Vemos en las figuras 4.1-4.6 como al disminuir el valor de los pesos (se ha tomado el mismo valor del peso para todos los puntos), la función spline se pega más a los puntos de control.

Como se puede apreciar en la figura 4.6 si se cumple que $\rho_i = 0$ para todo i, entonces $z_i = y_i$, y el spline suavizador resulta ser un spline interpolante. Esto se traduce en que cuanto mayor es la precisión con que están dadas las magnitudes y_i, tanto menores deben ser los coeficientes de peso respectivos. Si es necesario que el spline pase por el punto (x_k, y_k), el coeficiente de peso ρ_i ,$\forall\ i = 1, \dots, m$, correspondiente se deba hacer igual a cero. Sin embargo si se cumple que $\rho_i = \infty$, el ajuste se convierte en una recta de regresión. Con esto queda ilustrado como la bondad del ajuste depende del parámetro ρ.

En los cálculos prácticos lo más importante es la elección de los números ρ_i. Notar que los pesos pueden ser diferentes para puntos distintos. Así por ejemplo en el caso práctico naval que expondremos exigiremos que $\rho_1 = \rho_m = 0$, es decir, que los pesos en los extremos sean cero y así la curva pase por estos puntos, pero los pesos serán distintos de cero en los demás puntos de control.

Definimos $\Delta_i$como el error de medición de la magnitud y_i. Entonces es lógico exigir que

$$|S(x_i) - y_i| \leq \Delta_i$$

El caso más sencillo de elección de los coeficientes de peso ρ_i puede ser

$$\rho_i = c\Delta_i$$

donde c es cierta constante suficientemente pequeña.

4.6 Construcción de la función Spline Cúbico Suavizante.

En cada intervalo $[x_i, x_{i+1}]$,$\forall\ i = 1, \dots, m-1$, se busca un polinomio de grado 3 que satisfaga:

$$S_i(x_i) = z_i, \quad S_i(x_{i+1}) = z_{i+1}$$
$$S_i''(x_i) = \eta_i, \quad S_i''(x_{i+1}) = \eta_{i+1}$$

Se requiere también una condición de continuidad en la primera derivada, pero ésta será impuesta posteriormente. La expresión

$$S(x) = S_i(x) = z_i(1-t) + z_{i+1}t - \frac{h_i^2}{6}t(1-t)[(2-t)\eta_i + (1+t)\eta_{i+1}], \qquad (4.6)$$

ofrece un polinomio de grado 3 que cumple con las condiciones dichas.

Como comprobación, partimos de que la derivada segunda de un polinomio de grado 3 es un polinomio de grado 1, y en este caso debe de satisfacer $S_i''(x_i) = \eta_i, \quad S_i''(x_{i+1}) = \eta_{i+1}$, como ya se ha citado anteriormente. Así si consideramos que

$$t = \frac{x - x_i}{h_i} \quad \text{y} \quad x_{i+1} = x_i + h_i,$$

la derivada segunda del polinomio es:

$$S_i''(x) = (1-t)\eta_i + t\eta_{i+1}.$$

Si integramos el polinomio anterior y realizando un cambio de variable:

$$\int S_i''(x)dx = \int S_i''(t)h_i dt = h_i(-\eta_i \frac{(1-t)^2}{2} + \frac{t^2}{2}\eta_{i+1}) + C,$$

donde el cambio de variable ha sido,

$$\frac{x-x_i}{h_i} = t \rightarrow dx = h_i dt,$$

Si volvemos a integrar el polinomio:

$$S_i(x) = \int S_i'(x)dx = \int S_i'(t)h_i dt = \frac{h_i^2}{2}\int(-\eta_i(1-t)^2 + t^2\eta_{i+1})dt) + h_i tC + D,$$

$$S_i(x) = \frac{h_i^2}{6}[(1-t)^3\eta_i + t^3\eta_{i+1}] + h_i tC + D. \quad (4.7)$$

Para las condiciones establecidas; debe ser

$$x = x_i \rightarrow S_i(x_i) = z_i \text{ , con t=0,}$$

$$x = x_{i+1} \rightarrow S_i(x_{i+1}) = z_{i+1} \text{ , con t=1,}$$

$$\frac{h_i^2}{6}[\eta_i] + D = z_i \text{ -> } D=z_i-\frac{h_i^2}{6}[\eta_i],$$

al sustituir el valor de D,

$$\frac{h_i^2}{6}[\eta_{i+1}] + h_i C + D = z_{i+1} \text{ -> } h_i C=z_{i+1} - \frac{h_i^2}{6}[\eta_{i+1}] + z_i-\frac{h_i^2}{6}[\eta_i],$$

y por último C

$$C = \frac{z_{i+1}-z_i}{h_i} + \frac{h_i}{6}(\eta_i - \eta_{i+1}).$$

Sustituyendo los valores de las constantes de integración C y D de la ecuación 4.7,

$$h_i tC + D = (z_{i+1} - z_i)t + \frac{h_i^2}{6}(\eta_i - \eta_{i+1})t + z_i - \frac{h_i^2}{6}\eta_i.$$

Nuestro polinomio queda de la forma:

$$S_i(x) = \tfrac{h_i^2}{6}[(1-t)^3\eta_i + t^3\eta_{i+1}] + (z_{i+1} - z_i)t + \tfrac{h_i^2}{6}(\eta_i - \eta_{i+1})t + z_i - \tfrac{h_i^2}{6}\eta_i. \quad (4.8)$$

La forma lineal la podemos reescribir como

$$h_i tC + D - [z_i(1-t) + z_{i+1}t] + [z_i(1-t) + z_{i+1}t] =$$
$$= -\eta_{i+1}\frac{h_i^2}{6}t + \frac{h_i^2}{6}\eta_i t - \frac{h_i^2}{6}\eta_i + (1-t)z_i + t(z_i + 1).$$

Con las siguientes observaciones para los términos que acompañan a η_i

$$-t(1-t)(2-t) = -(t-t^2)(2-t) = -(2t - 2t^2 - t^2 + t^3) = -2t + 3t^2 - t^3 \text{ (I)}$$
$$(1-t)^3 = -t^3 + 3t^2 - 3t + 1, \text{ (II)}$$

y la diferencia de I y II,

$$(-2t + 3t^2 - t^3) - (-t^3 + 3t^2 - 3t + 1) = -t + 1,$$

sustituyendo:

$$(-t+1)\frac{h_i^2}{6}\eta_i. \quad (4.9)$$

Los términos que acompañan a $\boldsymbol{\eta_{i+1}}$ son

$$-t(1-t)(1+t) = -t(1-t^2) = -t + t^3 \text{ (I)}$$
$$t^3 \text{ (II)},$$

y la diferencia de I y II

$$(-t + t^3) - t^3 = t,$$

y de nuevo incluyendo los coeficientes

$$t\ \frac{h_i^2}{6}\eta_{i+1}. \tag{4.10}$$

Volviendo a la ecuación 4.8, y sustituyendo en ella las expresiones anteriores, 4.9 y 4.10, se obtiene el spline suavizante de la forma

$$S(x) = S_i(x) = z_i(1-t) + z_{i+1}t - \frac{h_i^2}{6}t(1-t)[(2-t)\eta_i + (1+t)\eta_{i+1}]. \tag{4.11}$$

4.7 Obtención de los coeficientes del Spline Cúbico Suavizante.

Vamos a trabajar en el intervalo

$$[x_i, x_{i+1}] \quad \forall \ \ i = 1, \dots, m-1.$$

Si introducimos las notaciones

$$S(x_i) = z_i, \quad S''(x_i) = \eta_i, \quad \forall \ \ i = 1,2, \dots, m,$$

donde z_i y η_i son $2m+2$ valores desconocidos.

Como ya se demostró anteriormente el spline cúbico suavizante se busca de la forma:

$$S(x) = S_i(x) = z_i(1-t) + z_{i+1}t - \frac{h_i^2}{6}t(1-t)[(2-t)\eta_i + (1+t)\eta_{i+1}],$$

donde

$$h_i = x_{i+1} - x_i\ , \qquad t = \frac{x-x_i}{h_i},$$

y los números z_i, η_i, i=0,1,…,m, son la solución del sistema de ecuaciones algebraicas lineales determinado por las condiciones impuestas.

La función $S_i(x)$ es continua en todo el intervalo [a,b]: para los dos primeros nodos, t tomará el valor t=0 y t=1 respectivamente, sustituyendo en la fórmula 4.11, obtenemos:

$$\boldsymbol{S_i(x_i) = z_i},$$
$$\boldsymbol{S_i(x_{i+1}) = z_{i+1}}.$$

Así los números z_i y η_i, deben ser elegidos de acuerdo a que el spline tenga derivada primera continua en el intervalo [a,b], para ello vamos a calcular la derivada primera de la función del spline suavizante, $S_i(x)$

En el en el intervalo $[x_{i-1}, x_i]$:

$$S'(x) = S'_{i-1}(x) = \frac{-z_{i-1} + z_i}{h_{i-1}} - \frac{h_{i-1}}{6}[(2 - 6t + 3t^2)\eta_{i-1} + (1 - 3t^2)\eta_i].$$

En el punto $x_i - 0$ (para t=1), tendremos

$$S'(x_i - 0) = S'_{i-1}(x_i) = \frac{-z_{i-1}+z_i}{h_{i-1}} + \frac{h_{i-1}}{6}[\eta_{i-1} + 2\eta_i]. \qquad (4.12)$$

Calculamos la derivada primera del spline el en el intervalo $[x_i, x_{i+1}]$:

$$S'(x) = S'_i(x) = \frac{-z_i + z_{i+1}}{h_i} - \frac{h_i}{6}[(2 - 6t + 3t^2)\eta_i + (1 - 3t^2)\eta_{i+1}].$$

En el punto $x_i + 0$ (para t=0) tendremos

$$S'(x_i + 0) = S_i'(x_i) = \frac{-z_i + z_{i+1}}{h_i} - \frac{h_i}{6}(2\eta_i + \eta_{i+1}). \quad (4.13)$$

Con la condición de continuidad para la primera derivada del spline en los puntos interiores del retículo ω,

$$S'(x_i - 0) = S'(x_i + 0), \qquad \forall \qquad i = 1, \dots, m-1,$$

obtendremos m-1 relaciones tal que

$$\frac{-z_{i-1} + z_i}{h_{i-1}} + \frac{h_{i-1}}{6}[\eta_{i-1} + 2\eta_i] = \frac{-z_i + z_{i+1}}{h_i} - \frac{h_i}{6}(2\eta_i + \eta_{i+1}),$$
$$\frac{-z_i + z_{i+1}}{h_i} + \frac{z_{i-1} - z_i}{h_{i-1}} = \frac{h_{i-1}}{6}(\eta_{i-1} + 2\eta_i) + \frac{h_i}{6}(2\eta_i + \eta_{i+1}).$$

Simplificando

$$\frac{z_{i-1}}{h_{i-1}} - \left(\frac{1}{h_i} + \frac{1}{h_{i-1}}\right) z_i + \frac{z_{i+1}}{h_i} = \frac{1}{6}(h_{i-1}\eta_{i-1} + 2(h_{i-1} + h_i)\eta_i + h_i\eta_{i+1}), \quad (4.14)$$

obteniendo m-1 ecuaciones.

De la condición de continuidad de la primera derivada del spline en los puntos interiores del retículo ω,

$$\frac{z_1}{h_1}-\left(\frac{1}{h_2}+\frac{1}{h_1}\right)z_2+\frac{z_3}{h_2}=\frac{1}{6}[h_1\eta_1+2(h_1+h_2)\eta_2+h_2\eta_3],$$

$$\vdots$$

$$\frac{z_{i-1}}{h_{i-1}}-\left(\frac{1}{h_i}+\frac{1}{h_{i-1}}\right)z_i+\frac{z_{i+1}}{h_i}=\frac{1}{6}[h_{i-1}\eta_{i-1}+2(h_{i-1}+h_i)\eta_i+h_i\eta_{i+1}],$$

$$\vdots$$

$$\frac{z_{m-2}}{h_{m-2}}-\left(\frac{1}{h_{m-2}}+\frac{1}{h_{m-1}}\right)z_{m-1}+\frac{z_m}{h_{m-1}}=\frac{1}{6}[h_{m-2}\eta_{m-2}+2(h_{m-2}+h_{m-1})\eta_{m-1}+h_{m-1}\eta_m],$$

$$\forall \qquad i=2,\dots,m-1.$$

Pudiendo obtenerse la siguiente relación matricial:

$$A\eta=6Hz, \tag{4.15}$$

donde:

$$A=\begin{bmatrix} h_1 & 2(h_1+h_2) & h_2 & 0 & \dots & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \end{bmatrix}$$

$$\mathrm{H}=\begin{bmatrix} \frac{1}{h_1} & (-\frac{1}{h_1}-\frac{1}{h_2}) & \frac{1}{h_2} & 0 & \dots & 0 \\ 0 & \frac{1}{h_2} & (-\frac{1}{h_2}-\frac{1}{h3}) & \frac{1}{h_3} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \frac{1}{h_{m-2}} & (-\frac{1}{h_{m-2}}-\frac{1}{h_{m-1}}) & \frac{1}{h_{m-1}} \end{bmatrix}$$

Quedando el sistema de ecuaciones incompleto, aún faltan ecuaciones.

Para la condición de minimización del *funcional* $J(S) = \int_a^b (S''(x))^2 dx \ + \ \sum_{i=0}^{m} \frac{1}{\rho i} (S(x_i) - y_i)^2$, calculamos el primer término

$$\int_a^b (S''(x))^2 dx \ = \int_0^1 [(1-t)\eta_i + t\eta_{i+1}]^2 h_i dt$$

$$t = \frac{x - x_i}{h_i} \rightarrow dx = h_i dt$$

Habiendo realizado el cambio de variable y desarrollando el binomio,

$$h_i \int_0^1 (1-t)^2 {\eta_i}^2 + t^2 {\eta_{i+1}}^2 + 2t(1-t)\eta_i \eta_{i+1} dt,$$

e integrando la expresión anterior,

$$\left[h_i \left[-\frac{(1-t)^3}{3} \right]_0^1 {\eta_i}^2 + \left[\frac{t^3}{3} \right]_0^1 {\eta_{i+1}}^2 + \left[t^2 - \frac{2t^3}{3} \right]_0^1 \eta_i \eta_{i+1} \right] = h_i \left[\frac{1}{3} {\eta_i}^2 + \frac{1}{3} {\eta_{i+1}}^2 + \frac{1}{3} \eta_i \eta_{i+1} \right].$$

Así obtenemos el primer miembro de la expresión del funcional:

$$\int_a^b (S''(x))^2 dx \ = h_i \left[\tfrac{1}{3} {\eta_i}^2 + \tfrac{1}{3} {\eta_{i+1}}^2 + \tfrac{1}{3} \eta_i \eta_{i+1} \right]. \qquad (4.16)$$

Vamos a desarrollar el funcional $J(s)$ mediante la técnica de los multiplicadores de Lagrange, la variable auxiliar λ_1, impondrá que se satisfaga la condición de contorno en el extremo izquierdo. La variable λ_m por su parte impondrá que se cumpla la condición de contorno del extremo derecho. Los valores λ_i , $i = 2, \dots, m-1$ impondrán que se satisfagan las condiciones debidas a la continuidad de la primera derivada.

Derivando el funcional respecto de λ , $\frac{dJ_a(s)}{d\lambda}$, se obtiene el sistema de ecuaciones

$$\tilde{\mathrm{A}}\eta = 6\hat{\mathrm{H}}z + F, \tag{4.17}$$

donde $\tilde{\mathrm{A}}, \hat{\mathrm{H}}$ y F , dependerán de las condiciones de contorno elegidas.

Despejando η en la ecuación 4.17:

$$\eta = \tilde{\mathrm{A}}^{-1}(6\,\ddot{\mathrm{H}}z + F). \tag{4.18}$$

donde necesitamos que $\tilde{\mathrm{A}}$ sea invertible.

Derivando el funcional auxiliar J_a respecto de η, $\frac{dJ_a(s)}{d\eta}$, obtenemos m ecuaciones más, que expresadas matricialmente quedan:

$$U\,\eta = V\boldsymbol{\lambda}. \tag{4.19}$$

Al despejar λ en la ecuación 4.18**,**

$$\lambda = V^{-1}U\eta, \tag{4.20}$$

donde nuevamente pedimos que V sea invertible.

Sustituyendo el valor anterior de η de la ecuación 4.18 en la ecuación 4.20,

$$\lambda = V^{-1}U\tilde{A}^{-1}(6\ddot{H}z + F). \tag{4.21}$$

Al definir $M = V^{-1}U\tilde{A}^{-1}$, y sustituyendo:

$$\lambda = 6M\ddot{H}z + MF. \tag{4.22}$$

Derivando el funcional respeto de z, obtenemos

$$z = Y + \frac{1}{2}\rho\, J\lambda. \tag{4.23}$$

Y al sustituir el valor de λ de la ecuación 4.22:

$$z = Y + \frac{1}{2}\rho\, J\,(6M\ddot{H}z + MF) = Y + \frac{1}{2}K_1 z + K_{2,} \tag{4.24}$$

donde K_1, K_2son dos matrices auxiliares, introducidas para simplificar la notación del sistema resultante,

$$K_1 = \rho\, JM6\ddot{H},$$
$$K_2 = \rho\, JMF.$$

Y definiendo una tercera matriz auxiliar C

$$C = Id - \frac{1}{2}K_1,$$

el sistema resultante quedará

$$Cz = Y + K_2. \tag{4.25}$$

Para poder resolver este sistema de ecuaciones es necesario definir las matrices anteriores, que vendrán dadas en función de las condiciones de contorno elegidas, las cuales ya se comentó que se elegirían en función de los datos adicionales que tengamos sobre el comportamiento de la función.

Notar que este sistema siempre tendrá solución única para cualquier valor de ρ salvo para un número finito de valores de ρ que tienen que ver con los valores propios de la matriz C. En particular, para valores de ρ suficientemente pequeños (caso que nos es más interesante en la práctica) siempre hay solución única.

4.8 El sistema de ecuaciones final en función de las condiciones de contorno.

A continuación se va a detallar cuáles serán las dos ecuaciones que se podrán obtener en función de las condiciones de contorno, las cuales incluso podrán ser distintas en cada extremo.

En caso de conocer condiciones de ***primer tipo***, es decir, los valores de la primera derivada de $S(x)$ en los extremos del intervalo $[a, b]$; se obtendrían las dos ecuaciones que nos faltan para completar el sistema del siguiente modo:

- Obtenemos la derivada del trozo de Spline correspondiente al extremo.
- Evaluamos esta ecuación en punto extremo.
- Y lo igualamos al valor conocido de la primera derivada.

1) Para el extremo izquierdo (a):

$$S_1'(x_1) = \frac{-z_1 + z_2}{h_1} - \frac{h_1}{6}(2\eta_1 + \eta_1) = f'(a),$$

$$\frac{-z_1 + z_2}{h_1} = f'(a) + \frac{h_1}{6}(2\eta_1 + \eta_1).$$

2) Para el extremo derecho (b):

$$S'_{m-1}(x_m) = \frac{-z_{m-1} + z_m}{h_{m-1}} + \frac{h_{m-1}}{6}[\eta_{m-1} + 2\eta_m] = f'(b),$$
$$\frac{-z_{m-1} + z_m}{h_{m-1}} = f'(b) - \frac{h_{m-1}}{6}[\eta_{m-1} + 2\eta_m]$$

Si utilizáramos condiciones de ***segundo tipo***, es decir, damos los valores de la segunda derivada en los extremos del intervalo $[a, b]$; básicamente se han de dar los valores de η_1 y η_m:

$$f''(a) = \eta_1\,, \quad f''(b) = \eta_m.$$

En caso de que la función a suavizar sea *periódica*, o de ***tercer tipo***,de periodo $T = b - a$, en particular, $f(a) = f(b)$ y por tanto $S_1(a) = S_{m-1}(b)$; se utilizarán condiciones de *tercer tipo*, las cuales darán las siguientes ecuaciones:

La primera ecuación adicional sería $S'_1(t_1) = S'_{m-1}(t_m)$, y sustituyendo:

$$\frac{-z_1 + z_2}{h_1} - \frac{h_1}{6}(2\eta_1 + \eta_1) = \frac{-z_{m-1} + z_m}{h_{m-1}} + \frac{h_{m-1}}{6}[\eta_{m-1} + 2\eta_m],$$
$$\frac{-z_1 + z_2}{h_1} - \frac{-z_{m-1} + z_m}{h_{m-1}} = \frac{h_{m-1}}{6}[\eta_{m-1} + 2\eta_m] + \frac{h_1}{6}(2\eta_1 + \eta_1).$$

La segunda ecuación obtenida sería $S''_1(t_1) = S''_{m-1}(t_m)$, en nuestro caso de suavización

$$\eta_1 - \eta_m = 0.$$

No siempre conocemos los mismos datos en ambos extremos del intervalo $[a, b]$, por lo que podemos tener diferentes condiciones en cada extremo, excepto en las condiciones de tercer tipo que exigen que la función sea periódica de periodo T=b-a y en particular $f(a) = f(b)$.

Por último antes de empezar a estudiar los diferentes casos en los que nos podremos encontrar a la hora de aproximar mediante splines suavizantes, recordamos que las condiciones de primer tipo tienen preferencia sobre las de segundo tipo, pudiendo aproximar estos valores.

A partir de aquí estudiaremos los casos más representativos, los cuales necesitan ser estudiados individualmente para poder demostrar que el sistema de ecuaciones resultante tiene solución y además es única, es decir vamos a demostrar que las matrices $\tilde{A}\, y\, V$ son invertibles en todos los casos.

Para estudiar la invertibilidad de dichas matrices vamos a hacer uso constantemente e un resultado que asegura que una matriz estrictamente diagonal dominante es invertible.

4.9 Comprobación del sistema compatible determinado.

Una vez construido un sistema de ecuaciones, es muy importante comprobar si el sistema es compatible determinado, con única solución, pudiendo obtenerse ésta y así poder obtener el spline correspondiente.

Si construimos el sistema completo $Az = B$ para que éste tenga solución compatible determinada sólo habrá que demostrar que la matriz A es invertible, es decir, que el determinante de A no es nulo.

En matemáticas, en particular en álgebra lineal, una matriz cuadrada *A* de orden *n* se dice que es **invertible**, **no singular**, **no degenerada** o **regular** si existe otra matriz cuadrada de orden *n*, llamada **matriz inversa** de *A* y representada como A^{-1}, tal que:

$$A \cdot A^{-1} = A^{-1}A = I_n,$$

donde I_n es la matriz identidad de orden n y el producto utilizado es el producto de matrices usual.

Existe un resultado que asegura que todo matriz estrictamente diagonal dominante es invertible.

Formalmente, se dice que la matriz A de orden n es *estrictamente diagonal dominante* cuando se satisface:

$$|a_{i,i}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{i,j}| \qquad \forall\, i = 1, \dots, n.$$

El resultado, conocido como lema de Hadamard, se prueba de la siguiente manera:

Si la matriz A no fuese invertible, la ecuación $Az = 0$ admitiría una solución no nula. Entonces existe:

$$z = [z_{1,}, \dots \dots, z_n]^T,$$

una solución. Podemos suponer sin pérdida de generalidad que

$$\max_{1\leq i\leq n} |z_i| = 1.$$

Sea r un índice para el que es $|z_r| = 1$. Tomando módulos en la ecuación

$$a_{r1}z_1 + \cdots + a_{rr}z_r + \cdots + a_{rn}z_n = 0.$$

Se concluye que

$$|a_{rr}| \leq \sum_{i\neq r} |a_{r1}||z_1| \leq \sum_{i\neq r} |a_{r1}|.$$

Desigualdad que contradice la hipótesis de que la matriz A es estrictamente diagonal dominante. Esto permite dar por demostrado que una matriz estrictamente diagonal dominante es invertible.

4.10 El sistema de ecuaciones final y comprobación de que el sistema es compatible determinado para cada caso particular.

En este último apartado nos vamos a centrar en buscar el sistema de ecuaciones final para los casos particulares más representativos (los demás casos se deducen fácilmente a partir de éstos) y además demostraremos que cada uno de los sistemas que se forman, incluidas las condiciones de contorno, son compatibles determinados lo cual nos garantizará que la función spline se pueda construir.

Para abreviar se utilizarán las siguientes denominaciones:

1: condiciones de primer tipo (se conoce el valor de la primera derivada en el extremo).
2: condiciones de segundo tipo (se conoce el valor de la segunda derivada en el extremo).
3: condiciones de tercer tipo (existe periocidad).
CI: condición de contorno en el extremo izquierdo.
CD: condición de contorno en el extremo derecho.

4.10.1 Caso CI=1, CD=1.

Las condiciones en los extremos son de primer tipo, por lo que se conocen las primeras derivadas y tenemos las dos siguientes ecuaciones adicionales:

$$\frac{-z_1 + z_2}{h_1} = f'(a) + \frac{h_1}{6}(2\eta_1 + \eta_2),$$

$$\frac{-z_{m-1} + z_m}{h_{m-1}} = f'(b) - \frac{h_{m-1}}{6}[\eta_{m-1} + 2\eta_m].$$

Al desarrollar el funcional para las condiciones de contorno establecidas

$$J_a(S) = h_1\left[\tfrac{1}{3}{\eta_1}^2 + \tfrac{1}{3}{\eta_2}^2 + \tfrac{1}{3}\eta_1\eta_2\right] + \cdots + h_{m-1}\left[\tfrac{1}{3}{\eta_{m-1}}^2 + \tfrac{1}{3}{\eta_m}^2 + \tfrac{1}{3}\eta_{m-1}\eta_m\right] + \frac{1}{\rho_1}(z_1 - y_1)^2 + \cdots + \frac{1}{\rho_m}(z_m - y_m)^2 + \lambda_1\left(\frac{-z_1+z_2}{h_1} - \frac{h_1}{6}(2\eta_1 + \eta_2) - f'(a)\right) + \lambda_2\left[\left(\frac{z_1}{h_1} - \left(\frac{1}{h_2} + \frac{1}{h_1}\right)\right)z_2 + \frac{z_3}{h_2} - \frac{1}{6}(h_1\eta_1 + 2(h_1 + h_2)\eta_2 + h_2\eta_3)\right] + \cdots + \lambda_i\left[\left(\frac{z_{i-1}}{h_{i-1}} - \left(\frac{1}{h_i} + \frac{1}{h_{i-1}}\right)\right)z_i + \frac{z_{i+1}}{h_i} - \frac{1}{6}(h_{i-1}\eta_{i-1} + 2(h_{i-1} + h_i)\eta_i + h_i\eta_{i+1})\right] + \cdots + \lambda_{m-1}\left[\left(\frac{z_{m-2}}{h_{m-2}} - \left(\frac{1}{h_{m-2}} + \frac{1}{h_{m-1}}\right)\right)z_{m-1} + \frac{z_m}{h_{m-1}} - \frac{1}{6}(h_{m-2}\eta_{m-2} + 2(h_{m-2} + h_{m-1})\eta_{m-1} + h_{m-1}\eta_m)\right] + \lambda_m\left(\frac{-z_{m-1}+z_m}{h_{m-1}} + \frac{h_{m-1}}{6}[\eta_{m-1} + 2\eta_m] - f'(b)\right),$$

$$\forall \quad i = 3, \ldots, m-2.$$

Operando el funcional $\frac{dJ_a(s)}{d\lambda}$ se obtiene el sistema de ecuaciones inicial ampliado

$$\tilde{\mathrm{A}}\eta = 6\hat{\mathrm{H}}z + F,$$

donde las matrices,

$$\tilde{\mathrm{A}} = \begin{bmatrix} \frac{h_1}{3} & \frac{h_1}{6} & 0 & 0 & \cdots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \ddots & \vdots \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \\ 0 & 0 & 0 & 0 & -\frac{h_{m-1}}{6} & -\frac{h_{m-1}}{3} \end{bmatrix}$$

$$\hat{H}=\begin{bmatrix} -\frac{1}{h_1} & \frac{1}{h_1} & 0 & 0 & 0 & 0 \\ \frac{1}{h_1} & (-\frac{1}{h_1}-\frac{1}{h_2}) & \frac{1}{h_2} & 0 & \cdots & \vdots \\ 0 & \frac{1}{h_2} & (-\frac{1}{h_2}-\frac{1}{h3}) & \frac{1}{h_3} & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & \frac{1}{h_{m-2}} & (-\frac{1}{h_{m-2}}-\frac{1}{h_{m-1}}) & \frac{1}{h_{m-1}} \\ 0 & 0 & 0 & 0 & -\frac{1}{h_{m-1}} & \frac{1}{h_{m-1}} \end{bmatrix}$$

$$F=\begin{bmatrix} f'(a) \\ 0 \\ \vdots \\ 0 \\ f'(b) \end{bmatrix}$$

Como se puede comprobar fácilmente la matriz A es estrictamente diagonal dominante y por tanto será invertible.

Realizando la siguiente operación al funcional $\frac{dJ_a(s)}{d\eta}$, obtenemos m ecuaciones más,

$$\frac{2}{3}h_1\eta_1+\frac{h_1}{3}\eta_2-\frac{h_1}{3}\lambda_1-\frac{h_1}{6}\lambda_2=0,$$

$$\frac{h_1}{3}\eta_1+\frac{2}{3}(h_1+h_2)\eta_2+\frac{h_2}{3}\eta_3-\frac{h_1}{6}\lambda_1-\frac{1}{3}(h_1+h_2)\lambda_2-\frac{h_2}{6}\lambda_3=0,$$

$$\vdots$$

$$\frac{h_i}{3}\eta_{i-1}+\frac{2}{3}(h_{i-1}+h_i)\eta_i+\frac{h_i}{3}\eta_{i+1}-\frac{1}{3}(h_{i-1}+h_i)\lambda_i-\frac{h_i}{6}\lambda_{i+1}=0,$$

$$\vdots$$

$$\frac{h_{m-3}}{3}\eta_{m-3}+\frac{2}{3}(h_{m-3}+h_{m-2})\eta_{m-2}+\frac{h_{m-2}}{3}\eta_{m-1}-\frac{1}{3}(h_{m-3}+h_{m-2})\lambda_{m-2}-\frac{h_{m-3}}{6}\lambda_{m-2}=0,$$

$$\frac{h_{m-2}}{3}\eta_{m-2}+\frac{2}{3}(h_{m-2}+h_{m-1})\eta_{m-1}+\frac{h_{m-1}}{3}\eta_m-\frac{1}{3}(h_{m-2}+h_{m-1})\lambda_{m-1}-\frac{h_{m-2}}{6}\lambda_{m-1}+\frac{h_{m-1}}{6}\lambda_m=0,$$

$$\frac{2}{3}h_{m-1}\eta_m+\frac{h_{m-1}}{3}\eta_{m-1}+\frac{h_{m-1}}{3}\lambda_m-\frac{h_{m-1}}{6}\lambda_{m-1}=0,$$

$$\forall \quad i=3,\dots,m-1.$$

Que también puede ser escrito matricialmente como

$$U\,\eta = V\lambda,$$

donde

$$\mathrm{U}=\begin{bmatrix} \frac{2h_1}{3} & \frac{h_1}{3} & 0 & 0 & \dots & 0 \\ \frac{h_1}{3} & \frac{2(h_1+h_2)}{3} & \frac{h_2}{3} & 0 & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \frac{h_{m-2}}{3} & \frac{2(h_{m-2}+h_{m-1})}{3} & \frac{h_{m-1}}{3} \\ 0 & 0 & 0 & 0 & \frac{h_{m-1}}{3} & \frac{2h_{m-1}}{3} \end{bmatrix}$$

$$\mathrm{V}=\begin{bmatrix} \frac{h_1}{3} & \frac{h_1}{6} & 0 & 0 & \dots & 0 \\ \frac{h_1}{6} & \frac{(h_1+h_2)}{3} & \frac{h_2}{6} & 0 & \ddots & 0 \\ 0 & \frac{h_2}{6} & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \frac{h_{m-2}}{6} & \frac{2(h_{m-2}+h_{m-1})}{3} & -\frac{h_{m-1}}{6} \\ 0 & 0 & 0 & 0 & \frac{h_{m-1}}{6} & -\frac{h_{m-1}}{3} \end{bmatrix}$$

La matriz V se observa fácilmente que es estrictamente diagonal dominante, y por lo tanto es invertible.

Derivando el funcional respecto de los z_i, $\frac{dJ_a(s)}{dz}$,

$$\frac{2}{\rho_1}(z_1 - y_1) - \frac{\lambda_1}{h_1} + \frac{\lambda_2}{h_1} = 0,$$
$$\frac{2}{\rho_2}(z_2 - y_2) + \frac{\lambda_1}{h_1} - \left(\frac{1}{h_1} + \frac{1}{h_2}\right)\lambda_2 + \frac{\lambda_3}{h_2} = 0,$$
$$\vdots$$
$$\frac{2}{\rho_{i+1}}(z_i - y_i) - \left(\frac{1}{h_{i-1}} + \frac{1}{h_i}\right)\lambda_i + \frac{\lambda_{i+1}}{h_i} = 0,$$
$$\vdots$$
$$\frac{2}{\rho_{m-2}}(z_{m-2} - y_{m-2}) - \left(\frac{1}{h_{m-3}} + \frac{1}{h_{m-2}}\right)\lambda_{m-2} + \frac{\lambda_{m-3}}{h_{m-3}} = 0,$$
$$\frac{2}{\rho_{m-1}}(z_{m-1} - y_{m-1}) - \frac{\lambda_m}{h_{m-1}} - \left(\frac{1}{h_{m-2}} + \frac{1}{h_{m-1}}\right)\lambda_{m-1} + \frac{\lambda_{m-2}}{h_{m-2}} = 0,$$
$$\frac{2}{\rho_m}(z_m - y_m) + \frac{\lambda_m}{h_{m-1}} + \frac{\lambda_{m-1}}{h_{m-1}} = 0,$$
$$\forall \quad i = 3, \dots, m-1.$$

Donde despejando z,

$$z_1 = y_1 + \frac{\rho_1}{2}\left(+\frac{\lambda_1}{h_1} - \frac{\lambda_2}{h_1}\right),$$
$$z_2 = y_2 + \frac{\rho_2}{2}\left(-\frac{\lambda_1}{h_1} + \left(\frac{1}{h_1} + \frac{1}{h_2}\right)\lambda_2 - \frac{\lambda_3}{h_2}\right),$$
$$\vdots$$
$$z_i = y_i + \frac{\rho_i}{2}\left(\left(\frac{1}{h_{i-1}} + \frac{1}{h_i}\right)\lambda_i - \frac{\lambda_{i+1}}{h_i}\right),$$
$$\vdots$$
$$z_{m-2} = y_{m-2} + \frac{\rho_{m-2}}{2}\left(\left(\frac{1}{h_{m-3}} + \frac{1}{h_{m-2}}\right)\lambda_{m-2} - \frac{\lambda_{m-3}}{h_{m-3}}\right),$$
$$z_{m-1} = y_{m-1} + \frac{\rho_{m-1}}{2}\left(\frac{\lambda_m}{h_{m-1}} + \left(\frac{1}{h_{m-2}} + \frac{1}{h_{m-1}}\right)\lambda_{m-1} - \frac{\lambda_{m-2}}{h_{m-2}}\right),$$
$$z_m = y_m + \frac{\rho_m}{2}\left(-\frac{\lambda_{m-1}}{h_{m-1}} - \frac{\lambda_m}{h_{m-1}}\right),$$
$$\forall \quad i = 3, \dots, m-1.$$

Matricialmente puede ser expresado como,

$$Z = Y + \frac{1}{2}\rho \, \mathrm{J} \, \lambda,$$

donde,

$$\mathrm{J}=\begin{bmatrix} \frac{1}{h_1} & -\frac{1}{h_1} & 0 & 0 & \dots & 0 \\ -\frac{1}{h_1} & \frac{1}{h_1}+\frac{1}{h_2} & -\frac{1}{h_2} & \ddots & \ddots & \vdots \\ 0 & \frac{1}{h_2} & \frac{1}{h_2}+\frac{1}{h_3} & \ddots & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & -\frac{1}{h_{m-2}} & 0 \\ \vdots & \ddots & \ddots & -\frac{1}{h_{m-2}} & \frac{1}{h_{m-2}}+\frac{1}{h_{m-1}} & \frac{1}{h_{m-1}} \\ 0 & \dots & 0 & 0 & -\frac{1}{h_{m-1}} & -\frac{1}{h_{m-1}} \end{bmatrix}$$

4.10.2 Caso CI=2, CD=2.

Las condiciones en los extremos son de segundo tipo, por lo que se conocen las segundas derivadas y tenemos las dos siguientes ecuaciones adicionales:

$$f''(a) = \eta_1,$$
$$f''(b) = \eta_m.$$

Al desarrollar el funcional para las condiciones de contorno establecidas

$$J_a(s) = h_1\left[\frac{1}{3}{\eta_1}^2 + \frac{1}{3}{\eta_2}^2 + \frac{1}{3}\eta_1\eta_2\right] + \cdots + h_{m-1}\left[\frac{1}{3}{\eta_{m-1}}^2 + \frac{1}{3}{\eta_m}^2 + \frac{1}{3}\eta_{m-1}\eta_m\right] + \frac{1}{\rho_1}(z_1 - y_1)^2 + \cdots + \frac{1}{\rho_m}(z_m - y_m)^2 + \lambda_1(\eta_1 - f''(a)) + \cdots + \lambda_i\left[\left(\frac{z_{i-1}}{h_{i-1}} - \left(\frac{1}{h_i} + \frac{1}{h_{i-1}}\right)\right)z_i + \frac{z_{i+1}}{h_i} - \frac{1}{6}(h_{i-1}\eta_{i-1} + 2(h_{i-1} + h_i)\eta_i + h_i\eta_{i+1})\right] + \cdots + \lambda_{m-1}\left[\left(\frac{z_{m-2}}{h_{m-2}} - \left(\frac{1}{h_{m-2}} + \frac{1}{h_{m-3}}\right)\right)z_{m-2} + \frac{z_m}{h_{m-1}} - \frac{1}{6}(h_{m-2}\eta_{m-2} + 2(h_{m-2} + h_{m-1})\eta_{m-1} + h_{m-1}\eta_m)\right] + \lambda_m(\eta_m - f''(b))$$

$$\forall \quad i = 2, \ldots, m-1.$$

Operando el funcional $\frac{dJ_a(s)}{d\lambda}$ se obtiene el sistema de ecuaciones ampliado

$$\tilde{\mathrm{A}}\eta = 6\hat{\mathrm{H}}z + F,$$

donde

$$\tilde{\mathrm{A}} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \ddots & \vdots \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\hat{\mathrm{H}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{h_1} & (-\frac{1}{h_1} - \frac{1}{h_2}) & \frac{1}{h_2} & 0 & \cdots & \vdots \\ 0 & \frac{1}{h_2} & (-\frac{1}{h_2} - \frac{1}{h_3}) & \frac{1}{h_3} & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & \frac{1}{h_{m-2}} & (-\frac{1}{h_{m-2}} - \frac{1}{h_{m-1}}) & \frac{1}{h_{m-1}} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$F=\begin{bmatrix} f''(a) \\ 0 \\ \vdots \\ 0 \\ f''(b) \end{bmatrix}$$

Y como se puede observar con facilidad la matriz A estrictamente diagonal dominante, y por lo tanto invertible.

Realizando la siguiente operación al funcional $\frac{dJ_a(s)}{d\eta}$, obtenemos m ecuaciones más,

$$\frac{2}{3}h_1\eta_1 + \frac{h_1}{3}\eta_2 + \lambda_1 - \frac{h_1}{6}\lambda_2 = 0,$$

$$\vdots$$

$$\frac{h_i}{3}\eta_{i-1} + \frac{2}{3}(h_{i-1}+h_i)\eta_i + \frac{h_i}{3}\eta_{i+1} - \frac{1}{3}(h_{i-1}+h_i)\lambda_i - \frac{h_i}{6}\lambda_{i+1} = 0,$$

$$\vdots$$

$$\frac{h_i}{3}\eta_{m-2} + \frac{2}{3}(h_{m-2}+h_{m-1})\eta_{m-1} + \frac{h_{m-1}}{3}\eta_m - \frac{1}{3}(h_{m-2}+h_{m-1})\lambda_{m-1} - \frac{h_{m-2}}{6}\lambda_{m-1} = 0,$$

$$\frac{2}{3}h_{m-1}\eta_m + \frac{h_{m-1}}{3}\eta_{m-1} + \lambda_m - \frac{h_{m-1}}{6}\lambda_{m-1} = 0,$$

$$\forall \quad i = 2, \dots, m-1.$$

Matricialmente puede ser expresado como

$$U\,\eta = V\lambda,$$

donde

$$U=\begin{bmatrix} \frac{2h_1}{3} & \frac{h_1}{3} & 0 & 0 & \dots & 0 \\ \frac{h_1}{3} & \frac{2(h_1+h_2)}{3} & \frac{h_2}{3} & 0 & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \frac{h_{m-2}}{3} & \frac{2(h_{m-2}+h_{m-1})}{3} & \frac{h_{m-1}}{3} \\ 0 & 0 & 0 & 0 & \frac{h_{m-1}}{3} & \frac{2h_{m-1}}{3} \end{bmatrix}$$

$$V=\begin{bmatrix} -1 & \frac{h_1}{6} & 0 & 0 & \dots & 0 \\ 0 & \frac{(h_1+h_2)}{3} & \frac{h_2}{6} & 0 & \ddots & 0 \\ 0 & \frac{h_2}{6} & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \frac{h_{m-2}}{6} & \frac{2(h_{m-2}+h_{m-1})}{3} & 0 \\ 0 & 0 & 0 & 0 & \frac{h_{m-1}}{6} & -1 \end{bmatrix}$$

Siendo la matriz V invertible, ya que su determinante es distinto de cero, como se comprueba fácilmente de desarrollar dicho determinante por la primera columna y posteriormente por la última. La matriz a la que se llega resulta ser estrictamente diagonal dominante y por tanto invertible y por tanto con determinante distinto de cero.

Derivando el funcional respecto de los z_i, $\frac{dJ_a(s)}{dz}$,

$$\frac{2}{\rho_1}(z_1-y_1)+\frac{\lambda_2}{h_1}=0,$$
$$\vdots$$
$$\frac{2}{\rho_i}(z_i-y_i)-(\frac{1}{h_{i-1}}+\frac{1}{h_i})\lambda_i+\frac{\lambda_{i+1}}{h_i}=0,$$
$$\vdots$$
$$\frac{2}{\rho_{m-1}}(z_{m-1}-y_{m-1})-(\frac{1}{h_{m-2}}+\frac{1}{h_{m-1}})\lambda_{m-1}+\frac{\lambda_{m-2}}{h_{m-2}}=0,$$
$$\frac{2}{\rho_m}(z_m-y_m)+\frac{\lambda_{m-1}}{h_{m-1}}=0,$$
$$\forall \quad i=2,\dots,m-1.$$

Despejando z,

$$z_1 = y_1 + \frac{\rho_1}{2}\left(-\frac{\lambda_2}{h_1}\right),$$

$$\vdots$$

$$z_i = y_i + \frac{\rho_i}{2}\left(\left(\frac{1}{h_{i-1}} + \frac{1}{h_i}\right)\lambda_i - \frac{\lambda_{i+1}}{h_i}\right),$$

$$\vdots$$

$$z_{m-1} = y_{m-1} + \frac{\rho_{m-1}}{2}\left(\left(\frac{1}{h_{m-2}} + \frac{1}{h_{m-1}}\right)\lambda_{m-1} - \frac{\lambda_{m-2}}{h_{m-2}}\right),$$

$$z_m = y_m + \frac{\rho_m}{2}\left(-\frac{\lambda_{m-1}}{h_{m-1}}\right),$$

$$\forall \quad i = 2, \dots, m-1.$$

Matricialmente,

$$Z = Y + \frac{1}{2}\rho \, \mathrm{J} \, \lambda,$$

donde,

$$\mathrm{J} = \begin{bmatrix} 0 & -\frac{1}{h_1} & 0 & 0 & \dots & 0 \\ 0 & \frac{1}{h_1} + \frac{1}{h_2} & -\frac{1}{h_2} & \ddots & \ddots & \vdots \\ 0 & \frac{1}{h_2} & \frac{1}{h_2} + \frac{1}{h_3} & \ddots & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & -\frac{1}{h_{m-2}} & 0 \\ \vdots & \ddots & \ddots & \frac{1}{h_{m-2}} & \frac{1}{h_{m-2}} + \frac{1}{h_{m-1}} & 0 \\ 0 & \dots & 0 & 0 & \frac{1}{h_{m-1}} & 0 \end{bmatrix}.$$

4.10.3 Caso CI=3, CD=3

Para este caso las ecuaciones adicionales se obtienen considerando que la función a suavizar es periódica de periodo $T = b - a$, de modo que aplicando las condiciones de tercer tipo se obtendrán las siguientes ecuaciones:

$$\eta_1 - \eta_m = 0,$$
$$6\left(\frac{-z_1 + z_2}{h_1} - \frac{-z_{m-1} + z_m}{h_{m-1}}\right) = h_1(2\eta_1 + \eta_2) + h_{m-1}[\eta_{m-1} + 2\eta_m].$$

Y necesitamos que los datos cumplan las condiciones de compatibilidad

$$\rho_1 = \rho_m,$$
$$y_1 = y_m.$$

Al desarrollar el funcional para las condiciones de contorno establecidas:

$$J_a(S) = h_1\left[\tfrac{1}{3}{\eta_1}^2 + \tfrac{1}{3}{\eta_2}^2 + \tfrac{1}{3}\eta_1\eta_2\right] + \cdots + h_{m-1}\left[\tfrac{1}{3}{\eta_{m-1}}^2 + \tfrac{1}{3}{\eta_m}^2 + \tfrac{1}{3}\eta_{m-1}\eta_m\right] + \frac{1}{\rho_1}(z_1 - y_1)^2 + \cdots + \frac{1}{\rho_m}(z_m - y_m)^2 + \lambda_1((\eta_1 - \eta_m)) + \lambda_2\left[\left(\frac{z_1}{h_1} - \left(\frac{1}{h_2} + \frac{1}{h_1}\right)\right)z_2 + \frac{z_3}{h_2} - \frac{1}{6}(h_1\eta_1 + 2(h_1 + h_2)\eta_2 + h_2\eta_3)\right] + \lambda_{i+1}\left[\left(\frac{z_i}{h_i} - \left(\frac{1}{h_{i+1}} + \frac{1}{h_i}\right)\right)z_{i+1} + \frac{z_{i+2}}{h_{i+1}} - \frac{1}{6}(h_i\eta_i + 2(h_i + h_{i+1})\eta_{i+1} + h_{i+1}\eta_{i+2})\right] + \cdots + \lambda_{m-1}\left[\left(\frac{z_{m-2}}{h_{m-2}} - \left(\frac{1}{h_{m-2}} + \frac{1}{h_{m-3}}\right)\right)z_{m-2} + \frac{z_m}{h_{m-1}} - \frac{1}{6}(h_{m-2}\eta_{m-2} + 2(h_{m-2} + h_{m-1})\eta_{m-1} + h_{m-1}\eta_m)\right] + \lambda_m\left(\frac{-z_1+z_2}{h_1} - \frac{-z_{m-1}+z_m}{h_{m-1}} - \frac{h_1}{6}(2\eta_1 + \eta_2) - \frac{h_{m-1}}{6}[\eta_{m-1} + 2\eta_m]\right) + \lambda_{m+1}(z_1 - z_m)$$

$$\forall \quad i = 2,3, \dots, m-3.$$

Operando el funcional $\frac{dJ_a(s)}{d\lambda}$ se obtiene el sistema de ecuaciones:

$$\tilde{\mathrm{A}}\eta = 6\ddot{\mathrm{H}}z + F,$$

donde

$$\tilde{A}=\begin{bmatrix} 1 & 0 & 0 & 0 & \dots & -1 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \ddots & \vdots \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \\ 2h_1 & h_1 & \dots & 0 & h_{m-1} & 2h_{m-1} \end{bmatrix}$$

$$\hat{H}=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{h_1} & (-\frac{1}{h_1}-\frac{1}{h_2}) & \frac{1}{h_2} & 0 & \dots & \vdots \\ 0 & \frac{1}{h_2} & (-\frac{1}{h_2}-\frac{1}{h3}) & \frac{1}{h_3} & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & \frac{1}{h_{m-2}} & (-\frac{1}{h_{m-2}}-\frac{1}{h_{m-1}}) & \frac{1}{h_{m-1}} \\ -\frac{1}{h_1}-\frac{1}{h_{m-1}} & \frac{1}{h_1} & 0 & \dots & \frac{1}{h_{m-1}} & 0 \end{bmatrix}$$

La matriz $\tilde{A}$ es invertible ya que su determinante es distinto de cero. Esto se comprueba sumando a la última columna la primera y desarrollando el determinante por la primera fila, ya que se llega a una matriz estrictamente diagonal dominante que como se sabe tiene determinante distinto de cero.

Realizando la siguiente operación al funcional $\frac{dJ_a(s)}{d\eta}$, obtenemos m ecuaciones más

$$\frac{2}{3}h_1\eta_1 + \frac{h_1}{3}\eta_2 + \lambda_1 - \frac{h_1}{6}\lambda_2 - \lambda_m\frac{h_1}{3} = 0,$$

$$\frac{h_1}{3}\eta_1 + \frac{2}{3}(h_1+h_2)\eta_2 + \frac{h_2}{3}\eta_3 - \frac{1}{3}(h_1+h_2)\lambda_2 - \frac{h_2}{6}\lambda_3 - \lambda_m\frac{h_1}{6} = 0,$$

$$\vdots$$

$$\frac{h_{i-1}}{3}\eta_{i-1} + \frac{2}{3}(h_{i-1}+h_i)\eta_i + \frac{h_i}{3}\eta_{i+1} - \frac{h_{i-1}}{6}\lambda_{i-1} - \frac{1}{3}(h_{i-1}+h_i)\lambda_i - \frac{h_i}{6}\lambda_{i+1} = 0,$$

$$\vdots$$

$$\frac{h_{m-2}}{3}\eta_{m-2} + \frac{2}{3}(h_{m-2}+h_{m-1})\eta_{m-1} + \frac{h_{m-1}}{3}\eta_m - \frac{h_{m-2}}{6}\lambda_{m-2} - \frac{1}{3}(h_{m-2}+h_{m-1})\lambda_{m-1} - \lambda_m\frac{h_{m-1}}{6} = 0,$$

$$-\lambda_1 + \frac{2}{3}h_{m-1}\eta_m + \frac{h_{m-1}}{3}\eta_{m-1} - \frac{h_{m-1}}{6}\lambda_{m-1} - \lambda_m\frac{h_{m-1}}{3} = 0,$$

$$\forall \quad i = 3, \dots, m-2.$$

Matricialmente puede ser expresado como

$$U\,\eta = V\lambda,$$

donde

$$\mathrm{U}=\begin{bmatrix} \frac{2h_1}{3} & \frac{h_1}{3} & 0 & 0 & \dots & 0 \\ \frac{h_1}{3} & \frac{2(h_1+h_2)}{3} & \frac{h_2}{3} & 0 & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \frac{h_{m-2}}{3} & \frac{2(h_{m-2}+h_{m-1})}{3} & \frac{h_{m-1}}{3} \\ 0 & 0 & 0 & 0 & \frac{h_{m-1}}{3} & \frac{2h_{m-1}}{3} \end{bmatrix}$$

$$V=\begin{bmatrix} -1 & \frac{h_1}{6} & 0 & \dots & 0 & \frac{h_1}{3} \\ 0 & \frac{(h_1+h_2)}{3} & \frac{h_2}{6} & 0 & \ddots & \frac{h_1}{6} \\ \vdots & \frac{h_2}{6} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \frac{h_{m-2}}{6} & \frac{2(h_{m-2}+h_{m-1})}{3} & \frac{h_{m-1}}{6} \\ 1 & 0 & 0 & 0 & \frac{h_{m-1}}{6} & \frac{h_{m-1}}{3} \end{bmatrix}$$

La matriz V es invertible ya que su determinante es distinto de cero. Esto se comprueba sumando a la última fila la primera y desarrollando el determinante por la primera columna, ya que se llega a una matriz estrictamente diagonal dominante que como se sabe tiene determinante distinto de cero.

Derivando el funcional, respecto de los z_i, $\frac{dJ_a(s)}{dz}$,

$$\frac{2}{\rho_1}(z_1-y_1)+\frac{\lambda_2}{h_1}-\frac{\lambda_m}{h_1}+\lambda_{m+1}=0,$$

$$\frac{2}{\rho_2}(z_2-y_2)-(\frac{1}{h_1}+\frac{1}{h_2})\lambda_2+\frac{\lambda_3}{h_2}+\frac{\lambda_m}{h_1}=0,$$

$$\vdots$$

$$\frac{2}{\rho_i}(z_i-y_i)+\lambda_{i-1}\frac{1}{h_{i-1}}-(\frac{1}{h_{i-1}}+\frac{1}{h_i})\lambda_i+\frac{\lambda_{i+1}}{h_i}=0,$$

$$\vdots$$

$$\frac{2}{\rho_{m-1}}(z_{m-1}-y_{m-1})-(\frac{1}{h_{m-2}}+\frac{1}{h_{m-1}})\lambda_{m-1}+\frac{\lambda_{m-2}}{h_{m-2}}+\lambda_m\frac{1}{h_{m-1}}=0,$$

$$\frac{2}{\rho_m}(z_m-y_m)+\frac{\lambda_{m-1}}{h_{m-1}}-\frac{\lambda_m}{h_{m-1}}-\lambda_{m+1}=0,$$

$$\forall \quad i=3,\dots,m-2.$$

De la primera y la última ecuación y de la condición $z_1 = z_m$, se obtiene el sistema de ecuaciones

$$\frac{2}{\rho_1}(z_1 - y_1) + \frac{\lambda_2}{h_1} - \frac{\lambda_m}{h_1} + \lambda_{m+1} = 0,$$
$$\frac{2}{\rho_m}(z_m - y_m) + \frac{\lambda_{m-1}}{h_{m-1}} - \frac{\lambda_m}{h_{m-1}} - \lambda_{m+1} = 0,$$
$$z_1 = z_m,$$

Restando a la primera ecuación la segunda, sustituyendo la tercera y teniendo en cuenta las condiciones de compatibilidad se obtiene el valor de λ_{m+1}

$$2\lambda_{m+1} = \frac{\lambda_2 - \lambda_m}{h_1} - \frac{\lambda_{m-1} - \lambda_m}{h_{m-1}},$$
$$\lambda_{m+1} = \frac{1}{2}\frac{\lambda_2 - \lambda_m}{h_1} - \frac{1}{2}\frac{\lambda_{m-1} - \lambda_m}{h_{m-1}}.$$

Sustituyendo este valor de λ_{m+1}en el sistema de ecuaciones obtenido derivando el funcional respecto de los z_i

$$\frac{2}{\rho_1}(z_1 - y_1) + \frac{3}{2}\frac{\lambda_2 - \lambda_m}{h_1} - \frac{1}{2}\frac{\lambda_{m-1} - \lambda_m}{h_{m-1}} = 0,$$
$$\frac{2}{\rho_2}(z_2 - y_2) - (\frac{1}{h_1} + \frac{1}{h_2})\lambda_2 + \frac{\lambda_3}{h_2} + \frac{\lambda_m}{h_1} = 0,$$
$$\vdots$$
$$\frac{2}{\rho_i}(z_i - y_i) + \lambda_{i-1}\frac{1}{h_{i-1}} - (\frac{1}{h_{i-1}} + \frac{1}{h_i})\lambda_i + \frac{\lambda_{i+1}}{h_i} = 0,$$
$$\vdots$$
$$\frac{2}{\rho_{m-1}}(z_{m-1} - y_{m-1}) - (\frac{1}{h_{m-2}} + \frac{1}{h_{m-1}})\lambda_{m-1} + \frac{\lambda_{m-2}}{h_{m-2}} + \lambda_m\frac{1}{h_{m-1}} = 0,$$
$$\frac{2}{\rho_m}(z_1 - y_m) + \frac{3}{2}\frac{\lambda_{m-1} - \lambda_m}{h_{m-1}} - \frac{1}{2}\frac{\lambda_2 - \lambda_m}{h_1} = 0,$$
$$\forall \quad i = 3, \dots, m-2.$$

Despejando z,

$$z_1 = y_1 - \frac{\rho_1}{2}\left(\frac{3}{2}\frac{\lambda_2 - \lambda_m}{h_1} - \frac{1}{2}\frac{\lambda_{m-1} - \lambda_m}{h_{m-1}}\right),$$
$$z_2 = y_2 - \frac{\rho_2}{2}\left(-\left(\frac{1}{h_1} + \frac{1}{h_2}\right)\lambda_2 + \frac{\lambda_3}{h_2} + \frac{\lambda_m}{h_1}\right),$$
$$\vdots$$
$$z_i = y_i - \frac{\rho_i}{2}\left(-\left(\frac{1}{h_{i-1}} + \frac{1}{h_i}\right)\lambda_i + \frac{\lambda_{i+1}}{h_i} + \frac{\lambda_{i-1}}{h_{i-1}}\right),$$
$$\vdots$$
$$z_{m-1} = y_{m-1} - \frac{\rho_{m-1}}{2}\left(-\left(\frac{1}{h_{m-2}} + \frac{1}{h_{m-1}}\right)\lambda_{m-1} + \frac{\lambda_{m-2}}{h_{m-2}} + \lambda_m\frac{1}{h_{m-1}}\right),$$
$$z_m = z_1 = y_m - \frac{\rho_m}{2}\left(\frac{3}{2}\frac{\lambda_{m-1} - \lambda_m}{h_{m-1}} - \frac{1}{2}\frac{\lambda_2 - \lambda_m}{h_1}\right),$$
$$\forall \quad i = 3, \dots, m-2.$$

Matricialmente puede expresarse como

$$Z = Y + \frac{1}{2}\rho\, \mathrm{J}\, \lambda,$$

donde

$$\mathrm{J} = \begin{bmatrix} 0 & -\frac{1}{2h_1} & 0 & \dots & -\frac{1}{h_{m-1}} & -\frac{1}{h_1} + \frac{1}{h_{m-1}} \\ 0 & \frac{1}{h_1} + \frac{1}{h_2} & -\frac{1}{h_2} & \ddots & \ddots & -\frac{1}{h_1} \\ 0 & \frac{1}{h_2} & \frac{1}{h_2} + \frac{1}{h_3} & \ddots & 0 & \vdots \\ 0 & \ddots & \ddots & \ddots & -\frac{1}{h_{m-2}} & \vdots \\ \vdots & \ddots & \ddots & \frac{1}{h_{m-2}} & \frac{1}{h_{m-2}} + \frac{1}{h_{m-1}} & -\frac{1}{h_{m-1}} \\ 0 & -\frac{1}{2h_1} & \dots & 0 & -\frac{1}{h_{m-1}} & \frac{1}{h_1} + \frac{1}{h_{m-1}} \end{bmatrix}.$$

4.10.4 Otros casos.

Existen otras posibles combinaciones además de las ya estudiadas en este capítulo, sin embargo, puesto que serán las mismas sólo que variando una de las dos ecuaciones frontera, se desarrollarán de igual modo demostrando que son matrices estrictamente diagonales dominantes y según el enunciado del lema de Hadamard serán por tanto, invertibles y su sistema tendrá solución compatible determinada.

Capítulo V
Resolución de Splines en Matlab

Capítulo V

Programación de Splines Cúbicos Suavizantes en Matlab

Para la programación de la función Spline, se ha elegido MATLAB, o también el denominado *''Laboratorio de matrices''* (abreviatura de *MATrix LABoratory*), que es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M).

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI), y la comunicación con programas en otros lenguajes y con otros dispositivos hardware.

Para la resolución de los splines cúbicos suavizantes, se ha creado una función capaz de resolver el algoritmo que se planteó y resolvió en el capítulo anterior, se ha tomado como punto de partida el caso de los spline cúbicos interpolantes desarrollado en un proyecto anterior por Irene Gallego Valdellós, (véase bibliografía). Para resolver dicho algoritmo en esta función tendremos que proporcionar una serie de *inputs*, y ésta a su vez nos devolverá una serie de *outputs,* la solución de nuestro problema.

Para el programa creado, ‘’Splines Cúbicos Suavizantes’’, tendremos que proporcionar las coordenadas, tanto de abscisas como de ordenadas, de los puntos donde se va a realizar la aproximación. En caso de que queramos evaluar la función spline en unos puntos determinados, habrá que facilitarle las abscisas de dichos puntos al programa.

Pero como no siempre utilizaremos esta evaluación, lo vamos a incluir como una opción. Por tanto como dato de entrada introduciremos $\mathrm{opx} = \text{'s'/'n'}$ para indicar si se calculan o no; y el retículo x, que serán puntos donde queremos evaluar la función polinómica a trozos.

Además de los nodos o puntos donde queremos realizar la aproximación y sus respectivos valores de la función, recordamos que estos datos no eran suficientes para poder llegar a la solución, se necesitaban dos ecuaciones adicionales, las cuales se obtenían de las condiciones de contorno en los extremos.
Para una aproximación mediante splines suavizantes, habrá que proporcionar al programa datos sobre las condiciones en los extremos izquierdo y derecho, CI y CD respectivamente. Por lo tanto proporcionaremos al programa los valores de la primera derivada, en caso de condiciones de contorno de primer tipo, de la segunda derivada para segundo tipo, y en caso de ser periódicas no necesitaremos dar datos adicionales.

Por último, habrá que proporcionar otro input, que será un vector con los coeficientes de peso o de suavizado, también conocido como ‘’weigth parameter’’, ρ. Como ya sabemos para valores pequeños de ρ, la función pasará por los puntos o cerca de éstos. Para valores altos de ρ, la función no pasará necesariamente por los nodos, como ya se demostró en el capítulo anterior.

Este valor del parámetro de suavizado puede ser dado por decisión propia del usuario, el cual podrá probar diferentes valores y decidir cual se ajusta mejor a sus necesidades, o incluso podrá elegirlo en función del grado de error o ruido que contengan sus mediciones de los nodos.

Como datos de salida u *outputs* se obtienen, aparte de los valores aproximados si es el caso, los polinomios de tercer grado que definirán el spline.

Asimismo se podrán visualizar las gráficas de la función spline, así como las funciones de la primera y segunda derivada, las cuales tienen un alto interés para verificar la pendiente o giro de la curva y las propiedades de *suavidad* del spline, pudiendo comprobar visualmente la uniformidad de la curvatura.

5.1 Scripts empleados.

En esta sección realizaremos una descripción de los scripts de Matlab que hemos utilizado para definir las matrices de coeficientes necesarias para resolver el sistema de ecuaciones. En este apartado incluiremos una descripción de los ficheros de datos que hemos utilizado. El programa con el que se ha realizado la resolución de splines cúbicos suavizantes, consiste en una serie de scripts de Matlab, junto con algunos archivos de datos en formato .txt, donde quedará reflejada la solución del sistema.

Para poder ejecutar dichos scripts es necesario que se encuentren todos en el directorio de trabajo de Matlab donde estamos trabajando, de no ser así el programa no funcionaría, es muy importante tener esto en cuenta.

En este apartado presentaremos los scripts que hemos desarrollado:

condicion_derivadasA.m

En este script se define la matriz de coeficientes A que acompaña a las η en el sistema de ecuaciones $\tilde{A}\eta = 6\ddot{H}z + F$, el cual se obtiene de la condición de continuidad de la primera derivada de la función spline cúbico suavizante.

```
function A=condicion_derivadas_A(h,CI,CD)

% Esta matriz serán los coeficientes que acompañan a los eta para
% condiciones en la frontera CI=1 y CD=1 es decir de segundo primer
% tipo en ambos extremos
% A es el término que acompaña a los eta tal que
% Ãeta=6Hz+F
% Este sistema de ecuaciones viene de la condición de continuidad de
% la
% primera derivada de la función Spline.
```

```
% Variables de entrada:
% h=vector h de distancias entre las abscisas
n=length(h);

% La primera ecuación
if CI==1
    A(1,1)=h(1)/3; A(1,2)=h(1)/6;
elseif CI==2
    A(1,1)=1;
elseif CI==3
    A(1,1)=1; A(1,n+1)=-1;
end

% Definimos el bucle que calculará los términos intermedios
for i=2:n
    A(i,i)=2*(h(i-1)+h(i));
    A(i,i+1)=h(i);
    A(i,i-1)=h(i-1);
end

% La última ecuación
if CD==1
    A(n+1,n)=-h(n)/6; A(n+1,n+1)=-h(n)/3;
elseif CD==2
    A(n+1,n+1)=1;
elseif CD==3
   A(n+1,1)=2*h(1); A(n+1,2)=h(1); A(n+1,n)=h(n); A(n+1,n+1)=2*h(n);
end
```

condicion_derivadasH.m

En este script se define la matriz de coeficientes H que acompaña a las z en el sistema de ecuaciones $\tilde{A}\eta = 6\ddot{H}z + F$, el cual se obtiene de la condición de continuidad de la primera derivada de la función spline cúbico suavizante.

```
function H=condicion_derivadas_H(h,CI,CD)

% Esta matriz serán los coeficientes que acompañan a los z para
% condiciones en la frontera CI=1 y CD=1 es decir de primer tipo en
ambos
% extremos
% H es el término que acompaña a los Z tal que
% Ãeta=6Hz+F
% Este sistema de ecuaciones viene de la condición de continuidad de
% la
% primera derivada de la función Spline.
% Variables de entrada
% h=vector h de distancias entre las abscisas

n=length(h);
H=zeros(n+1,n+1);
```

```
% La primera ecuación
if CI==1
    H(1,1)=-1/(6*h(1)); H(1,2)=1/(6*h(1));
elseif CI==2
    H(1,1)=0; H(1,2)=0;
elseif CI==4
    H(1,1)=1;

end

% Definimos el bucle que calculará los términos intermedios
for i=2:n
    H(i,i)=-(1/h(i-1)+1/h(i));
    H(i,i+1)=1/h(i);
    H(i,i-1)=1/h(i-1);
end

% La dos última ecuación
if CD==1
    H(n+1,n)=-1/(6*h(n)); H(n+1,n+1)=1/(6*h(n));
elseif CD==2
    H(n+1,n)=0; H(n+1,n+1)=0;
elseif CD==3
   H(n,n+1)=0; H(n,1)=1/h(n);
   H(n+1,1)=-1/h(1)-1/h(n); H(n+1,2)=1/h(1); H(n+1,n)=1/h(n);
H(n+1,n+1)=0;
elseif CD==4
   H(n+1,n+1)=1;
end
```

relacion_EtaLambda_U.m

En este script se define la matriz de coeficientes U que acompaña a las η en la ecuación $U\,\eta = V\lambda$. Esta ecuación se obtiene de la expresión desarrollada del funcional al diferenciar respecto de las η.

```
function U=relacion_EtaLambda_U(h)

% Matriz coeficientes obtenida de la expresión del funcional,
(dJ(s))/dEta)
% U*Eta=V*lambda
% U es la matriz que acompaña a lambda

% Variables de entrada:
% h=vector h de distancias entre las abscisas

% n es la dimensión del vector h
n=length(h);
```

```
% Definimos los términos invariantes
U(1,1)=2/3*h(1);
U(1,2)=h(1)/3;
U(n+1,n)=h(n)/3;
U(n+1,n+1)=2/3*h(n);

% Definimos el bucle que calculará el resto de términos
for i=2:n
    U(i,i)=2/3*(h(i-1)+h(i));
    U(i,i+1)=h(i)/3;
    U(i,i-1)=h(i-1)/3;
end
```

relacion_EtaLambda_V.m

En este script se define la matriz de coeficientes V que acompaña a los λ en la ecuación $U\,\eta = V\lambda$. Esta ecuación se obtiene de la minimización del funcional al diferenciar respecto de las η.

```
function V=relacion_EtaLambda_V(h,CI,CD)

% Matriz coeficientes obtenida de la expresión del funcional,
(dJ(s))/dEta)
% U*Eta=V*lambda
% V es la matriz q acompaña a lambda

% Variables de entrada:
% h=vector h de distancias entre las abscisas

% n es la dimensión del vector h
n=length(h);

% Las 2 primeras ecuaciones
if CI==1
    V(1,1)=h(1)/3; V(1,2)=h(1)/6;
    V(2,1)=h(1)/6; V(2,2)=(h(1)+h(2))/3; V(3,2)=h(2)/6;
elseif CI==2
    V(1,1)=-1; V(1,2)=h(1)/6;
    V(2,1)=0; V(2,2)=(h(1)+h(2))/3; V(3,2)=h(2)/6;

elseif CI==3
    V(1,1)=-1; V(1,2)=h(1)/6; V(1,n+1)=h(1)/3;
    V(2,1)=0; V(2,2)=1/3*(h(1)+h(2)); V(2,3)=h(2)/6; V(2,n+1)=h(1)/6;
elseif CI==4
    V(1,1)=0; V(1,2)=h(1)/6; V(1,n+1)=h(1)/3;
    V(2,1)=0; V(2,2)=1/3*(h(1)+h(2)); V(2,3)=h(2)/6; V(2,n+1)=h(1)/6;
end
```

```
% Definimos el bucle que calculará los términos intermedios
for i=3:n-1
    V(i,i)=(h(i-1)+h(i))/3;
    V(i-1,i)=h(i-1)/6;
    V(i+1,i)=h(i)/6;
end

% Las dos últimas ecuaciones
if CD==1
    V(n+1,n)=h(n)/6; V(n+1,n+1)=-h(n)/3;
    V(n,n-1)=h(n-1)/6; V(n,n)=(h(n-1)+h(n))/3; V(n,n+1)=-h(n)/6;
elseif CD==2
     V(n+1,n)=h(n)/6; V(n+1,n+1)=-1;
    V(n,n-1)=h(n-1)/6; V(n,n)=(h(n-1)+h(n))/3; V(n,n+1)=0;
elseif CD==3
    V(n+1,1)=1; V(n+1,n)=h(n)/6; V(n+1,n+1)=h(n)/3;
    V(n,n-1)=h(n-1)/6; V(n,n)=1/3*(h(n-1)+h(n)); V(n,n+1)=h(n)/6;
elseif CD==4
    V(n+1,1)=0; V(n+1,n)=h(n)/6; V(n+1,n+1)=0;
    V(n,n-1)=h(n-1)/6; V(n,n)=2/3*(h(n-1)+h(n)); V(n,n+1)=h(n)/6;

end
```

relacion_EtaLambda_J.m

En este script se define la matriz de coeficientes J, la cual se encuentra en la ecuación obtenida en el capítulo anterior $Z = Y + \frac{1}{2}\rho \, \mathrm{J} \, \lambda$.

Esta ecuación se obtiene de la minimización del funcional al diferenciar respecto de las z.

```
function J=relacion_ZetaLambda_J(h,CI,CD)

% Variables de entrada
% h=vector espaciado nodos
% De la minimización del funcional  y  (dJ(s))/dz
% se obtiene la relación
% Z=Y+1/2*rho*lambda*J
% En este programa definimos la matriz J
% n es la dimensión del vector h

n=length(h);
J=zeros(n+1,n+1);

% La primera ecuación
if CI==1
    J(1,1)=1/h(1); J(1,2)=-1/h(1);
    J(2,1)=-1/h(1); J(2,2)=1/h(1)+1/h(2); J(2,3)=-1/h(2);
elseif CI==2
    J(1,1)=0; J(1,2)=-1/h(1);
    J(2,1)=0; J(2,2)=1/h(1)+1/h(2); J(2,3)=-1/h(2);
elseif CI==3
    J(1,1)=0; J(1,2)=-1/2*1/h(1); J(1,n)=-1/2*1/h(n);
J(1,n+1)=1/2*1/h(1)+1/2*1/h(n);
    J(2,1)=0; J(2,2)=1/h(1)+1/h(2); J(2,3)=-1/h(2); J(2,n+1)=-1/h(1);
```

```
elseif CI==4
    J(1,1)=-1; J(1,2)=-1/h(1);
    J(2,1)=0; J(2,2)=1/h(1)+1/h(2); J(2,3)=-1/h(2);
end

% Definimos el bucle que calculará calculará los términos intermedios
for i=3:n-1
    J(i,i)=1/h(i-1)+1/h(i);
    J(i-1,i)=-1/h(i-1);
    J(i+1,i)=-1/h(i);
end

% La última ecuación
if CD==1
    J(n+1,n)=-1/h(n); J(n+1,n+1)=-1/h(n);
    J(n,n-1)=-1/h(n-1); J(n,n)=1/h(n-1)+1/h(n); J(n,n+1)=1/h(n);
elseif CD==2
    J(n+1,n)=-1/h(n); J(n+1,n+1)=0;
    J(n,n-1)=-1/h(n-1); J(n,n)=1/h(n-1)+1/h(n); J(n,n+1)=0;
elseif CD==3
     J(n,n-1)=-1/h(n-1); J(n,n)=1/h(n-1)+1/h(n); J(n,n+1)=-1/h(n);
     J(n+1,2)=-1/2*1/h(1); J(n+1,n)=-1/2*1/h(n);
J(n+1,n+1)=1/2*1/h(n)+1/2*1/h(1);
elseif CD==4
     J(n,n-1)=-1/h(n-1); J(n,n)=1/h(n-1)+1/h(n);
     J(n+1,n)=1/h(n); J(n+1,n+1)=-1;

end
```

5.2 Ficheros de resultados.

Estos ficheros serán los resultados obtenidos de la aproximación mediante splines cúbicos suavizantes, los ficheros que el usuario podrá visualizar son:

polinomios_splines.txt

En este archivo se incluye la fecha, día, mes y año, y la hora en la que se ejecutó el archivo. También se encuentra de manera simbólica el polinomio de tercer grado que define el spline en cada segmento.

resul_valores_aproximados.txt

En este archivo de nuevo se muestran además de la fecha y hora en que se ejecutó el programa, los resultados de los valores aproximados.

5.3 Fases del programa.

El programa Splines Cúbicos Suavizantes ha sido dividido en varias etapas perfectamente diferenciadas.

Primeramente es definida la función, en ella se indica el nombre del programa, las variables de entrada y las variables de salida. A continuación el usuario encontrará una serie de ayudas a las que podrá acceder mediante el comando ''help'' seguido del nombre del programa, en este caso ''splines_suavizantes''.

En una segunda etapa, son definidos una serie de errores que el usuario con pocos conocimientos sobre aproximación mediante splines puede encontrar muy útiles. Los errores con los que el usuario podrá encontrarse serán:

-En caso de que el usuario no introduzca al menos tres puntos o nodos de aproximación, no se podrá construir la función spline, por lo tanto el usuario será avisado mediante un mensaje de error.

-Si el usuario introdujera como condiciones en la frontera en alguno de los dos extremos condiciones de tercer tipo, implícitamente estará diciendo que en el extremo opuesto también serán de tercer tipo. En caso de que no sea así, el programa no funcionará y mostrará un mensaje de error.

En una tercera fase se definirán las matrices involucradas en los sistemas de ecuaciones lineales a resolver, las cuales como ya se ha explicado anteriormente dependen de las condiciones de contorno seleccionadas. Y se resuelven los sistemas obteniendo los coeficientes necesarios para la definición del spline.

En la cuarta fase del programa, se evaluarán los splines en los diferentes segmentos de acuerdo al polinomio obtenido en el capítulo anterior,

$$S(x) = S_i(x) = z_i(1-t) + z_{i+1}t - \frac{h_i^2}{6}t(1-t)[(2-t)\eta_i + (1+t)\eta_{i+1}].$$

Para concluir, se construirán de una manera simbólica los polinomios a partir de los coeficientes ya obtenidos. El programa dibujará mediante la función plot las diferentes gráficas de la función spline cúbico suavizante, así como su primera y segunda derivada. Además estos resultados, como ya se ha comentado, serán grabados en un fichero en formato .txt.

5.4 Código Matlab para la resolución de Splines Cúbicos Suavizantes.

```
function
Splines_Suavizantes(t,y,CI,S1a,S2a,CD,S1b,S2b,opx,x,opg1,opg2,opg3,opg
4,tol_rho)

% Esta función busca hallar las ecuaciones que definen los diferentes
% tramos que componen el spline
% suavizante y dibujar las gráficas de la curva, su primera y segunda
% derivada, así como los valores obtenidos al interpolar y su gráfica.
% Creando ficheros que recojan la información.
%
Splines_Suavizantes(t,y,CI,S1a,S2a,CD,S1b,S2b,opx,x,opg1,opg2,opg3,opg
4,tol_rho)
%
% Variables de entrada:
% t:abscisas de la tabla de valores de los nodos
% y:ordenadas de la tabla de valores de los nodos
% CI:condición de contorno en el extremo izquierdo
% CD:condición de contorno en  el extremo derecho

% Posibles condiciones de contorno:
% 1- De primer tipo: valores de la primera derivada
% 2- De segundo tipo: valores de la segunda derivada
% 3- De tercer tipo: periódicas de periodo T=b-a
%    S'(a)=S'(b); S''(a)=S''(b). Si CI=3->CD=3
% 4- De cuarto tipo: Tres veces diferenciable en los puntos t2 y/o tm-
1
%
% S1a:valor de la primera derivada en el extremo izquierdo
% S2a:valor de la segunda derivada en el extremo izquierdo
% S1b:valor de la primera derivada en el extremo derecho
% S2b:valor de la segunda derivada en el extremo derecho
%
% opx:parámetro que nos indica si debemos calcular o no los valores
interpolados
% 's':se calculan; 'n': no se calculan
% x:retículo de puntos donde queremos evaluar la función polinómica a
trozos
```

```
% opg1:parámetro que nos indica se debemos dibujar o no la gráfica de
los valoes interpolados
% opg2:parámetro que nos indica se debemos dibujar o no la gráfica de
los splines
% opg3:parámetro que nos indica se debemos dibujar o no la gráfica de
las primeras derivadas
% opg4:parámetro que nos indica se debemos dibujar o no la gráfica de
las segundas derivadas
% si opg1=1 entonces dibujará la gráfica
% tol_rho vector de tolerancias para los pesos del funcional, será el
que determine el
% grado de suavidad de la curva, es decir, que la curva realiza menos
oscilaciones
%
% EJEMPLO:
% t=[1,3,4,6]
% y=[2,-1,3,1]
% CI=2 ; CD=2
% S1a=0; S1b=0 (no se van a utilizar, da igual el dato)
% S2a=2; S2b=2
% opx='s'
% x=[1:0.2:6]
%
% Splines_Suavizantes([1,3,4,6],[2,-
1,3,1],2,0,0,2,0,0,'s',[1:0.2:6],1,1,1,1,[0.001,0.001,0.001,0.001]);

% Número de puntos en la tabla, longitud del vector nodos
n=length(t);
pe=length(tol_rho);
% Comprobamos que las entradas son correctas.
% El número mínimo de nodos para poder realizar un Spline es 3
if n<3
    uiwait(msgbox('El número de puntos de control debe ser mayor que
2', 'Mensaje de error',...
        'error','modal'));
    return;
% Si no se especifican las condiciones en la frontera, se avisará el
error
elseif CI==0 | CD==0
    uiwait(msgbox('Falta especificar alguna de las condiciones de
frontera', 'Mensaje de error',...
        'error','modal'));
    return;


% Si se marca condiciones periódicas o 3, va intrínseco que se tienen
% que dar en
% los tres exremos
elseif CI==3 & CD~=3
    uiwait(msgbox('Si CI=3 entonces CD debe ser también 3', 'Mensaje
de error',...
        'error','modal'));
    return;
elseif CD==3 & CI~=3
    uiwait(msgbox('Si CD=3 entonces CI debe ser también 3', 'Mensaje
de error',...
        'error','modal'));
    return;
```

```
elseif CD==3 & y(1)~=y(n)
    uiwait(msgbox('Para que la función sea periódica debe valer lo
mismo en los extremos', 'Mensaje de error',...
        'error','modal'));
    return;
% Con 3 nodos la condición CI=4 ya implica CD=4 y nos falta una
condición
elseif n==3 & CI==4 & CD==4
    uiwait(msgbox('Con 3 nodos la condición CI=4 ya implica CD=4 y nos
falta una condición', 'Mensaje de error',...
        'error','modal'));
    return;
elseif pe<3
     uiwait(msgbox('Se necesitan especificar al menos 3 coeficientes
de peso', 'Mensaje de error',...
        'error','modal'));
end


% Definimos el vector h de distancias entre las abscisas
h=diff(t,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%  Selección de las matrices  %%%%%%%%%%
%%%%%%  coeficiente en función     %%%%%%%%%%
%%%%%%  de las condiciones en la frontera %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Elegimos las matrices coeficiente en función de las condiciones de
contorno dadas
% Las matrices varían en función de las condiciones de la frontera
dato
% Elegiremos las matrices mediante un condicional

A=condicion_derivadas_A(h,CI,CD);
H=condicion_derivadas_H(h,CI,CD);
V=relacion_EtaLambda_V(h,CI,CD);
U=relacion_EtaLambda_U(h);
J=relacion_ZetaLambda_J(h,CI,CD);
F=relacion_EtaZeta_F(h,CI,CD,S1a,S1b,S2a,S2b);


% Definimos la matriz Auxiliar K1

K1_aux=6*J*V^(-1)*U*A^(-1)*H;
%rho=min(tol_rho,1./sum(abs(K1_aux)))
K1=diag(tol_rho)*K1_aux;

% Definimos la matriz Auxiliar K2
K2=diag(tol_rho)*J*inv(V)*U*inv(A)*F;

% Resolución del sistema lineal Cz=B
C=eye(n)-1/2*K1;
B=y+K2;
z=C\B;
```

```
% Cálculo de los valores de la segunda derivada en los nodos eta's

eta=A^(-1)*(6*H*z+F);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Evaluamos los Splines en x                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if opx=='s'

    % Encuentra el trozo polinómico a evaluar según cada entrada de x
    if x(1)==t(1)
        Si(1)=z(1);
        minim=Si; maxim=Si;
    else
        ind=find((x(1)>t)==0);
        ind=ind(1)-1;
        ind=ind(1);
        % Aplica la fórmula para calcular S_ind(x)
        tn=(x(1)-t(ind))/h(ind);
        Si(1)=z(ind)*(1-tn)+z(ind+1)*tn-h(ind)^2/6*tn*(1-tn)*((2-
tn)*eta(ind)+(1+tn)*eta(ind+1));
        minim=Si; maxim=Si;
    end

    for i=2:length(x)
        ind=find((x(i)>t)==0);
        ind=ind(1)-1;
        ind=ind(1);
        % Aplica la fórmula para calcular S_ind(x)
        tn=(x(i)-t(ind))/h(ind);
        Si(i)=z(ind)*(1-tn)+z(ind+1)*tn-h(ind)^2/6*tn*(1-tn)*((2-
tn)*eta(ind)+(1+tn)*eta(ind+1));
        if Si(i)<minim
            minim=Si(i);
        elseif Si(i)>maxim
            maxim=Si(i);
        end
    end

    if opg1==1
        figure(1);
        %Dibuja los puntos de control en la gráfica
        b1=plot(t,y,'ko','MarkerSize',2,'LineWidth',3);
        hold on;
        %Dibuja los valores interpolados en la gráfica
        b2=plot(x,Si,'ro','MarkerSize',3);
        hold on;
        axis([t(1)-0.1*(t(2)-t(1)) t(length(t))+...
            0.1*(t(2)-t(1)) min(min(y),minim)-...
            0.1*(max(y)-min(y)) max(max(y),maxim)+0.1*(max(y)-
min(y))]);
        axis equal;
        legend([b1,b2],'Puntos de Control','Valores Interpolados');
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Construimos los polinomios de la interpolación %
%                  Segmentaria                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

syms r;

%%%%% Establecemos una variable simbólica%%%%%%


for i=1:n-1
    S(i)=z(i)*(1-(r-t(i))/h(i))+z(i+1)*(r-t(i))/h(i)-...
        h(i)^2/6*(r-t(i))/h(i)*(1-(r-t(i))/h(i))*...
        ((2-(r-t(i))/h(i))*eta(i)+(1+(r-t(i))/h(i))*eta(i+1));
end


if opg2==1
    %Dibuja los puntos de control en la gráfica
    figure(2);
    b1=plot(t,y,'ko','MarkerSize',2,'LineWidth',3);
    hold on;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Dibuja cada uno de los polinomios%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    Ms=0;
    ms=0;
    for i=1:n-1
        b3=ezplot(S(i),[t(i),t(i+1)]);
        auxX=t(i):(t(i+1)-t(i))/10:t(i+1);
        auxY=subs(S(i),auxX); MauxY=max(auxY); mauxY=min(auxY);
        if MauxY>Ms
            Ms=MauxY;
        end
        if mauxY<ms
            ms=mauxY;
        end
    end
    axis([t(1)-0.1*(t(2)-t(1)) t(length(t))+...
        0.1*(t(2)-t(1))  ms-0.1*(Ms-ms)  Ms+0.1*(Ms-ms)]);
    axis equal;
    legend([b1,b3],'Puntos de Control','Splines Cúbicos');
    title('Ajuste por Splines Cúbicos Suavizantes');



    % dibuja los polinomio splines también en la interfaz gráfica
    axes(handles.grafica);


    b1=plot(t,y,'ko','MarkerSize',2,'LineWidth',3);
    hold on;
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Dibuja cada uno de los polinomios%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    for i=1:n-1
        b3=ezplot(S(i),[t(i),t(i+1)]);
    end
    axis([t(1)-0.1*(t(2)-t(1)) t(length(t))+...
        0.1*(t(2)-t(1))  ms-0.1*(Ms-ms)  Ms+0.1*(Ms-ms)]);
    axis equal;
    legend([b1,b3],'Puntos de Control','Splines Cúbicos');
    title('Ajuste por Splines Cúbicos Suavizantes');
    hold off

end


if opg3==1
    %Dibuja cada una de las primeras derivadas
    figure(3);
    hold on
    Ms=0;
    ms=0;

% Diferenciamos el polinomio construido, para obtener el polinomio de
la
% primera derivada.
    S1=diff(S,1);
    for i=1:n-1
        b4=ezplot(S1(i),[t(i),t(i+1)]);
        auxX=t(i):(t(i+1)-t(i))/10:t(i+1);
        auxY=subs(S1(i),auxX); MauxY=max(auxY); mauxY=min(auxY);
        if MauxY>Ms
            Ms=MauxY;
        end
        if mauxY<ms
            ms=mauxY;
        end
    end
    axis([t(1)-0.1*(t(2)-t(1)) t(length(t))+...
        0.1*(t(2)-t(1))  ms-0.1*(Ms-ms)  Ms+0.1*(Ms-ms)]);
    legend(b4,'Primera Derivada Splines Cúbicos');
    title('Primera Derivada de los Splines Cúbicos Suavizantes');
end


if opg4==1
    %Dibuja cada una de las segundas derivadas
    figure(4)
    hold on
```

```
% Diferenciamos el polinomio construido dos veces, para obtener el
polinomio de la
% segunda derivada.
    S2=diff(S,2);
    Msd=subs(S2(1),t(1));
    msd=subs(S2(1),t(1));
    for i=1:n-1
        b5=ezplot(S2(i),[t(i),t(i+1)]);
        aux=subs(S2(i),t(i+1));
        if aux>Msd
            Msd=aux;
        elseif aux<msd
            msd=aux;
        end
    end
    axis([t(1)-0.1*(t(2)-t(1)) t(length(t))+...
        0.1*(t(2)-t(1)) msd-0.1*(Msd-msd)  Msd+0.1*(Msd-msd)]);
    legend(b5,'Segunda Derivada Splines Cúbicos');
    title('Segunda Derivada de los Splines Cúbicos Suavizantes');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Salida de los polinomio a un fichero
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%  Se abre o crea un archivo y se escribe en él para los resultados de
valores aproximados

if opx=='s'
    fid=fopen('resul_valores_aproximados.txt','w');

fprintf(fid,'**********************************************************
**********************\n');
    hora=clock;
    fprintf(fid,'Resultado ejecutado el dia %d-%d-%d a las %d : %d :
%d\n',...
        hora(3),hora(2),hora(1),hora(4),hora(5),fix(hora(6)));

fprintf(fid,'**********************************************************
**********************\n\n');
    for i=1:length(x)
        fprintf(fid,'%f %f \n',x(i),Si(i));
    end
    fclose(fid);
end

%  Se abre o crea un archivo y se escribe en él para los resultados de
los polinomios del Spline

fid=fopen('polinomios_splines.txt','w');
fprintf(fid,'**********************************************************
**********************\n');
hora=clock;
fprintf(fid,'Resultado ejecutado el dia %d-%d-%d a las %d : %d :
%d\n',...
    hora(3),hora(2),hora(1),hora(4),hora(5),fix(hora(6)));
fprintf(fid,'**********************************************************
**********************\n\n');
signos='----';
```

```
for i=1:length(t)-1
    coefi=sym2poly(expand(S(i)));
    for j=1:4
        if coefi(j)>=0
            signos(j)='+';
        end
    end
    fprintf(fid,' %c %f x^3 %c %f x^2 %c %f x %c %f  \n',...
        signos(1),abs(coefi(1)),signos(2),abs(coefi(2)),signos(3),...
        abs(coefi(3)),signos(4),abs(coefi(4)));
end
fprintf(fid,'\n');
fclose(fid);
```

5.5 Ejemplo de ejecución del código en Matlab.

Para ejecutar el código creado, introducimos en la ventana de comandos o ''command window'' lo siguiente:

>>Splines_Suavizantes([1 3 4 6]',[2 -1 3 2]',1,2,0,1,2,0,'s',[1:0.2:6],1,1,1,1,[0.021, 0.021, 0.021, 0.021])

Esto quiere decir que vamos a realizar una aproximación mediante splines cúbicos suavizante en los puntos con valores de abcisas [1 3 4 6] y ordenadas [2 -1 3 2]. Las condiciones en la frontera izquierda serán de primer tipo, con un valor de la primera derivada igual a 2, y en la frontera derecha también serán de primer tipo con un valor de la primera derivada igual a 2. El intervalo donde se construirá el spline será [1,6]. Los siguientes cuatro valores indican que queremos que se nos muestren los gráficos de los puntos donde se realiza la aproximación, la función spline construida y la primera y segunda derivada de la función. Por último a los coeficientes de peso se les ha dado un valor de 0.021, permitiendo cierto grado de suavizado.

Los gráficos obtenidos fueron:

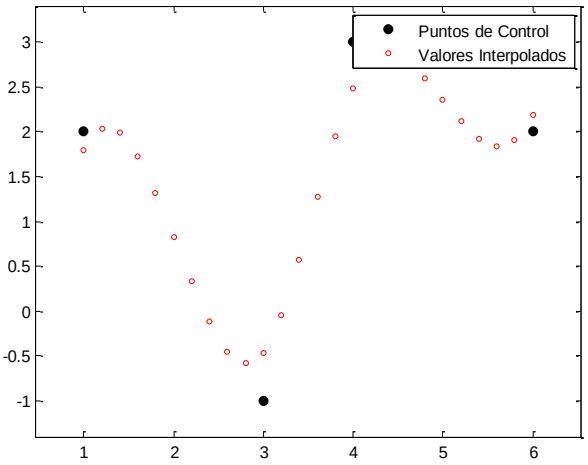


Figura 5.1: Puntos de control.

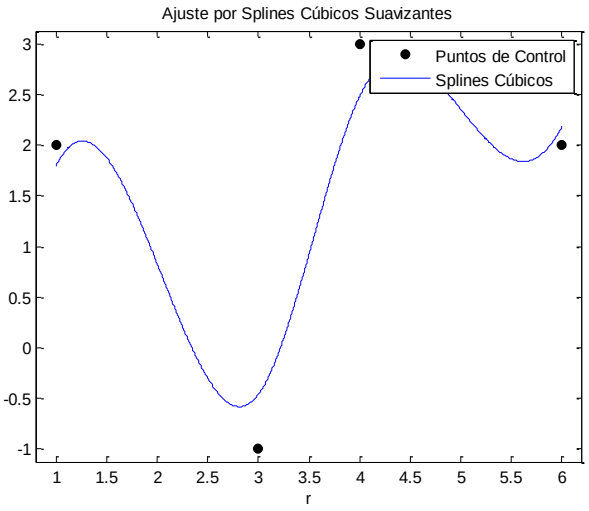


Figura 5.2: Función de ajuste mediante splines cúbicos suavizantes.

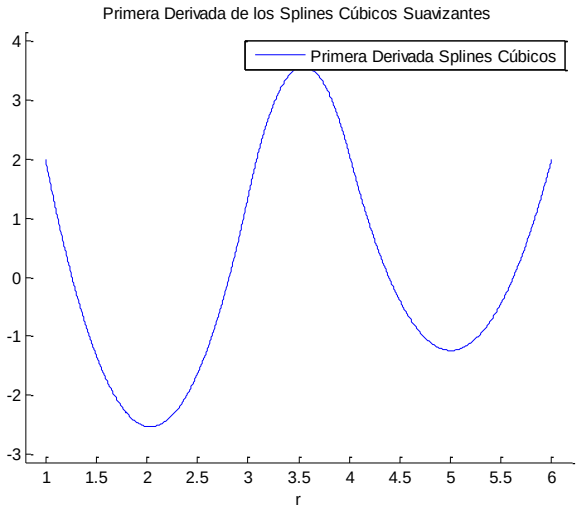


Figura 5.3: Primera derivada de la función spline.

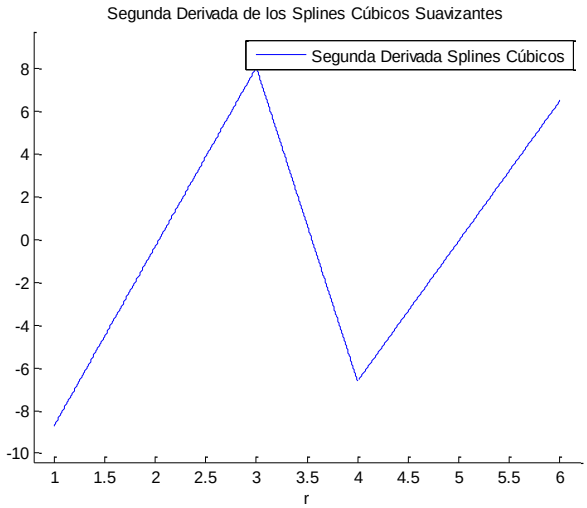


Figura 5.4: Segunda derivada de la función spline.

El resultado obtenido muestra una función spline en la cual efectivamente el valor de la primera derivada coincide con el valor de la primera derivada en los extremos introducido al ejecutar la función en matlab.

Los polinomios resultantes de la aproximación quedan reflejados en el archivo polinomios_splines.txt. Los polinomios resultantes son:

```
 + 1.401910 x^3 - 8.573740 x^2 + 14.941749 x - 5.976126=0
+ 2.450185 x^3 + 26.095119 x^2 + 89.064825 x + 98.030449=0
+ 1.095101 x^3 + 16.448314 x^2 + 81.108904 x + 128.867857=0
```

Capítulo VI

Interfaz Gráfica para la Resolución de Splines Cúbicos Suavizantes

Capítulo VI

Interfaz Gráfica para la Resolución de Splines Cúbicos Suavizantes

Para el desarrollo de la interfaz gráfica se ha utilizado de nuevo Matlab, para ello se eligió el entorno de trabajo de Matlab Guide (GUI), el cual es muy gráfico e intuitivo como podremos ver a continuación. Con esta interfaz se le proporcionará a cualquier usuario la posibilidad de poder resolver problemas de aproximación mediante el uso de éstos.

6.1 La interfaz gráfica GUI.

Matlab permite desarrollar fácilmente un conjunto de pantallas (paneles) con diferentes botones, menús, ventanas, etc., que permiten utilizar de manera muy simple e intuitiva programas realizados dentro de este entorno. Este conjunto de herramientas se denomina interfaz gráfica de usuario (GUI).

La elaboración de GUIs puede llevarse a cabo de dos formas, la primera de ellas consiste en escribir un programa que genere la GUI (script), la segunda opción consiste en utilizar la herramienta de diseño de GUIs, incluida en Matlab, llamada GUIDE (Graphical User Interfase Development Environment). Para este proyecto se ha decidido utilizar la propia herramienta de Matlab, GUIDE.

Con la documentación del presente Proyecto Fin Carrera se proporciona un CD de datos que contiene todo lo necesario para que cualquier usuario pueda ejecutar la interfaz gráfica "Splines Cúbicos Suavizantes". Lo único que necesitará será el programa informático MATLAB.

El contenido del CD adjunto contiene una carpeta llamada “Interfaz Gráfica”

Carpeta Splines Suavizantes: contiene todos los programas en formato .m que Matlab necesita para ejecutar el algoritmo de cálculo de los splines de forma correcta. No se podrá modificar bajo ningún concepto por el usuario, en caso de modificarse se puede correr el riesgo de que el programa no funcione y dé errores inesperados. Como excepción los dos únicos archivos dentro de esta carpeta que el usuario podrá modificar son:

- polinomios_splines.txt
- resul_valores_interpolados.txt

los cuales contienen los datos de salida del programa.

Archivos propios de la interfaz: en la carpeta principal se encuentran los archivos en formato .fig que Matlab necesita para poder ejecutar correctamente la interfaz. Estos archivos son:

- menú_ayuda.fig
- splines_cubicos_suavizantes.fig
- SplinesSuavizantes.fig

ArchivosAyuda: en la carpeta principal se han guardado todos los archivos en formato pdf que el programa reporta cuando se hace uso de la ayuda. Si el usuario lo desea puede leer cualquier archivo de ayuda. También se encontrará el tutorial sobre la interfaz. Los archivos que contiene son:

- Tutorial de la Interfaz gráfica.pdf
- Splines cúbicos suavizantes.pdf
- Resolución de Splines en Matlab.pdf

6.2 Ejecución de la Interfaz Gráfica.

Para ejecutar la interfaz gráfica, el usuario deberá realizar los siguientes pasos:

1. *Abrir* el programa MATLAB.

2. Seleccionar el *directorio de trabajo* de Matlab, es decir, la carpeta "Interfaz Gráfica". En la barra superior de Matlab debe aparecer, por ejemplo, la secuencia que se muestra en la figura siguiente:

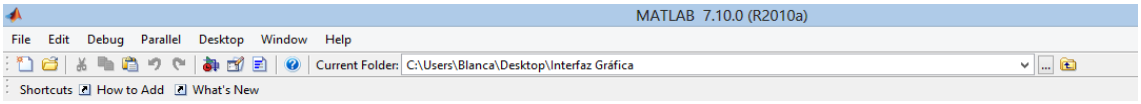


Figura 6.1: Directorio de trabajo de Matlab para ejecutar la Interfaz.

3. Escribir el nombre de la interfaz en la línea de comandos (pantalla principal de Matlab), como se muestra en la figura 6.2. En nuestro caso habrá que escribir ''*SplinesSuavizantes*''. Se recomienda que el usuario preste atención al cambio de letra mayúscula-minúscula, debido a que Matlab tiene en cuenta este aspecto.

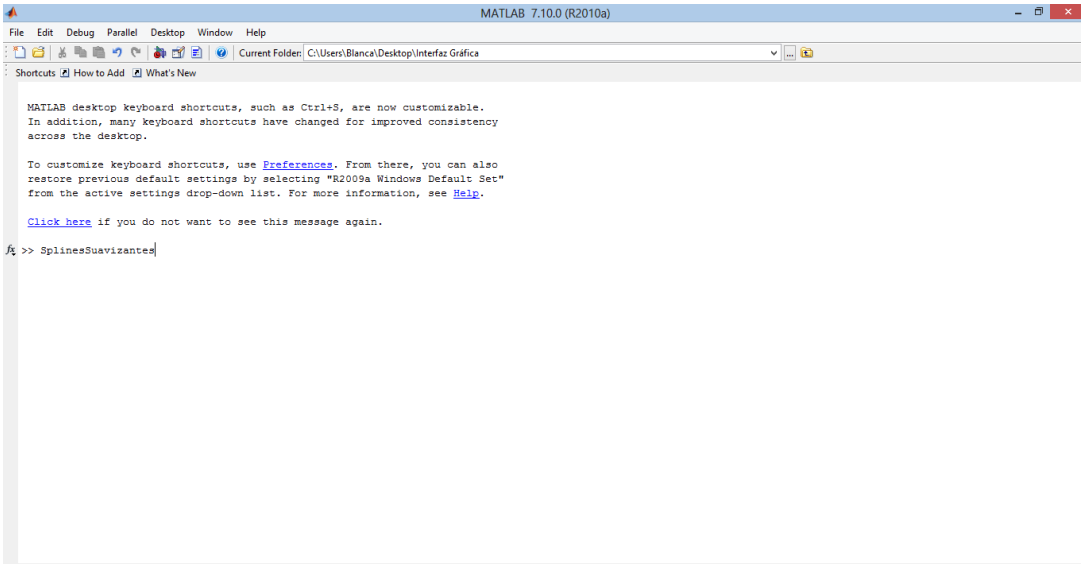


Figura 6.2: Ejecución de la interfaz gráfica

Una vez que el usuario accede a la pantalla inicial de la interfaz, figura 6.3, debe elegir entre realizar las opciones que se muestran.

Figura 6.3: Menú Principal de la Interfaz.

6.3 Pantalla Principal.

Al seleccionar en el menú principal la opción Comenzar, seremos redireccionados a la siguiente pantalla.

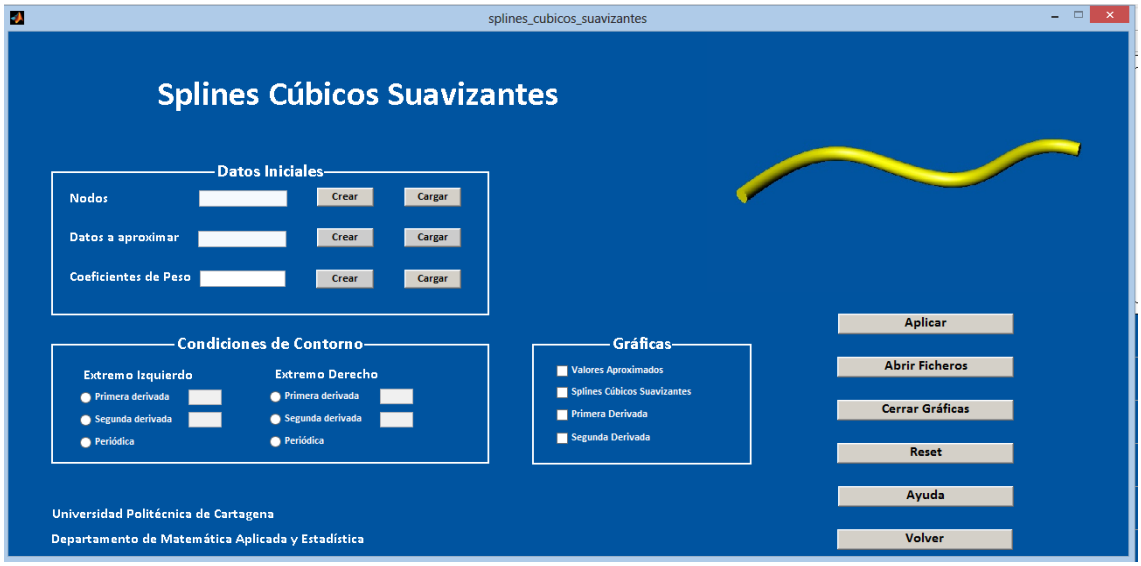


Figura 6.4: Pantalla principal de la interfaz gráfica.

Esta pantalla la hemos nombrado como la principal, ya que en ella es donde se realizará la aproximación mediante splines.

6.3.1 Panel de carga de datos iniciales.

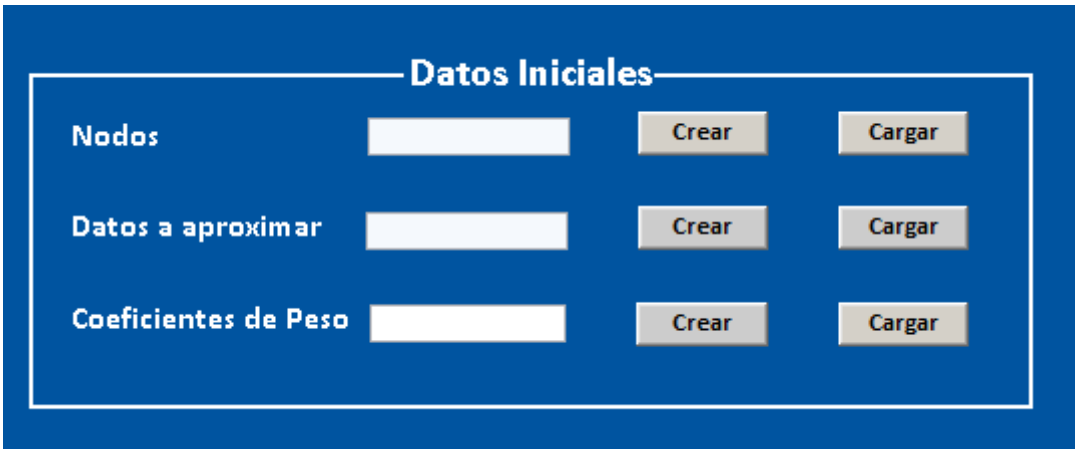


Figura 6.5: Panel de carga de los datos iniciales.

Nodos.

En este campo el usuario deberá introducir las coordenadas de los nodos donde se va a realizar la aproximación.

El usuario puede introducir estos valores mediante dos caminos distintos, crear o cargar.

- **Cargar**: Al pulsar el botón cargar el usuario accederá a un explorador con el que podrá seleccionar el archivo que contiene los valores de los nodos.
  El archivo creado es muy importante que se encuentre en la carpeta Interfaz Gráfica, y deberá tener la siguiente estructura.

```
function [t,y]=crear_nodos()


% completar vector de abscisas
t=[-6.65581,-12.3096,-15.4722,-18.284,-21.4086,-25.8184,-33.2787,-
34.8182,-42.521,-46.411,-48.8637,-51.3927,-54.0604,-57.0541];

% completar vector de ordenadas
y=[-1.83206e-16,1,2,3,4,5,5.54861,5.54861,5,4,3,2,1,0.253118];
```

- **Crear:** Al pulsar el botón Crear se abrirá una ventana del editor de Matlab con el archivo “crear_nodos.m” el cual podremos modificar para especificar las coordenadas de nuestros nodos. Recordamos que “t” son las abscisas e “y” las ordenadas.

Datos.

Para obtener los valores de las abcisas donde interpolar el spline construido, el procedimiento será análogo al que acabamos de explicar con los nodos. Pero en este caso el archivo que se abre en el editor de Matlab al pulsar el botón “Crear” correspondiente es “ crear_datos.m”, el script será el siguiente:

```
function x=crear_datos()

% completar las abscisas donde se quiere evaluar la reconstrucción

x=[0:0.5:1.5];
```

Coeficientes de peso.

En este campo el usuario introducirá los coeficientes de peso. El usuario podrá de nuevo tanto crear como cargar los archivos correspondientes. Los valores que definen los pesos pueden tener la siguiente estructura,

```
function p=crear_pesos()


% completar vector de pesos
p=[10^(-10),0.001*ones(1,12),10^(-10)];
```

donde p es el vector de pesos. En este caso se ha mantenido un valor de ρ=0.001 en el centro, y $\rho=10^{-10}$ en los extremos, es decir, los nodos de los extremos no tienen apenas libertad para ser aproximados.

6.3.2 Panel de carga de las condiciones de contorno.

En este panel el usuario tendrá diferentes opciones para definir las condiciones de contorno de su problema de aproximación.

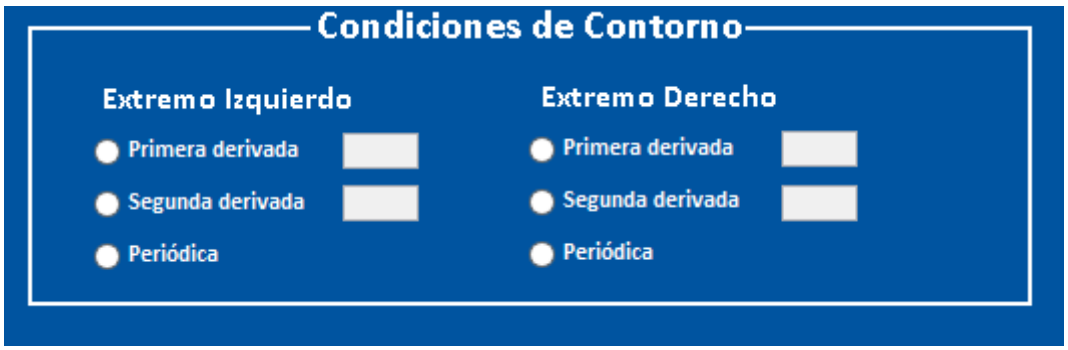


Figura 6.6: Panel de carga de las condiciones de contorno

Las opciones posibles son:

Primera derivada: Para condiciones de contorno de primer tipo, es necesario introducir el valor de la primera derivada del extremo en cuestión. Si se conociese este dato es aconsejable marcarlo y en ese caso se activará el cuadro de texto correspondiente donde se podrá escribir el valor conocido para la primera derivada.

Segunda derivada: Para condiciones de contorno de segundo tipo en el extremo derecho o izquierdo, será necesario introducir el valor de la segunda derivada. Igualmente si se conociese este valor, se aconseja marcar la opción segunda derivada activándose el cuadro de texto correspondiente, pudiendo escribir el valor de la segunda derivada en el extremo.

Periódica: En este caso nos referimos a condiciones de contorno de tercer tipo, o también llamadas periódicas. En caso de condiciones de contorno de tercer tipo se obliga automáticamente a que se den en los dos extremos.

Observaciones

- Para condiciones de contorno de *primer y segundo tipo*, se permite al usuario la posibilidad de utilizar condiciones distintas
- Al activar otro tipo de condición distinta de la periódica, en el panel se desactivarán las dos "Periódica", evitando así cometer el error de seleccionar que la curva es periódica sólo en un extremo, lo que no tiene sentido.
- Si por cualquier razón se quedara un cuadro de texto con algún valor numérico desactivado porque se ha decidido más tarde utilizar otro tipo de condición, no hay que preocuparse por él, puesto que no se usará este dato a la hora de construir el spline.
- Si podemos elegir entre aportar los valores de la primera o de la segunda derivada a la construcción del Spline, se aconseja especificar los valores de la primera derivada.

6.3.3 Panel de selección de las gráficas a mostrar.

En este panel podremos indicarle al programa que gráficas queremos que nos muestre al finalizar el cálculo de los splines cúbicos suavizantes.

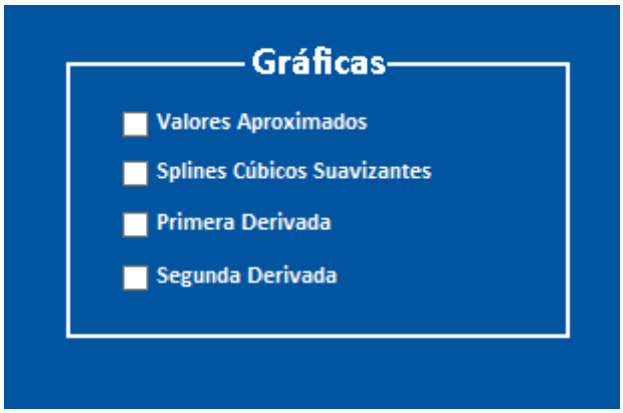


Figura 6.7: Panel de selección de gráficas

Las opciones posibles son:

Valores aproximados: mostrará una gráfica donde se muestran los valores del spline para las abscisas de aproximación introducidas, en caso de que este campo no esté vacío.
Splines Cúbicos Suavizantes: muestra el Spline construido, así como los nodos que se han introducido para su construcción.
Primera Derivada: al ser marcada mostrará la gráfica de la primera derivada de la curva construida.
Segunda Derivada: muestra la segunda derivada de la curva construida, muy útil para comprobar la uniformidad de la curvatura.

6.3.4 Opciones.

En la pantalla principal del programa se pueden encontrar las siguientes opciones:

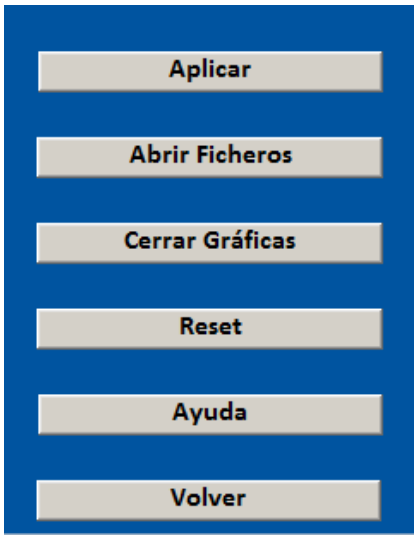


Figura 6.8: Opciones

- Aplicar: recoge los datos proporcionados en la interfaz gráfica y llama al programa de construcción de Splines ("Splines_Suavizantes.m") que se encarga de: la resolución del sistema, dibujar las gráficas que se han pedido y escribir los datos en los ficheros "polinomios_splines.txt" y "resul_valores_aproximados.m" si están creados (borrando los datos que existieran anteriormente) o los crea nuevos y escribe en ellos. Debido a que al presionar el botón "Aplicar" conlleva borrar los datos existentes anteriormente en el fichero con el nombre "polinomios_splines.txt" y, en caso de calcular valores interpolados, también en "resul_valores_aproximados.m"; es muy importante que guardemos los datos obtenidos en algún cálculo anterior de forma que al realizar nuevos cálculos no destruyamos los anteriores.

- Abrir Ficheros: abre en el editor de Matlab los ficheros creados en formato .txt, que contendrán los resultados de la aproximación.

- Cerrar Gráficas: cierra todas las gráficas que se hubieran abierto al aplicar el programa. Para volver a abrirlas es necesario volver a presionar el botón "Aplicar".

- Reset: borra todos los datos introducidos en la interfaz, incluidas las gráficas en caso de ya se hubiera ejecutado la interfaz anteriormente. Además desactiva todas las opciones marcadas. También borra la pantalla de comandos ("Command Window") de Matlab, pero no borra las variables que se hubieran definido previamente.

- **Ayuda**: este botón abrirá un archivo en formato .pdf que contiene este manual.

- Volver: este botón devuelve al usuario al menú principal del programa.

6.4 Menú Ayuda.

A través de esta pantalla, el usuario podrá consultar información acerca del funcionamiento de la interfaz. El aspecto general que presenta se muestra en la figura 6.9. Para consultar la información que alberga el menú ayuda, lo único que tiene que hacer el usuario es pulsar el botón correspondiente y esperar a que aparezca la información en un documento con formato .pdf (Acrobat Reader). A la pantalla mostrada en la figura 6.9 sólo se podrá acceder mediante el menú principal.

Figura 6.9: Menú Ayuda.

-Acerca de Splines Cúbicos Suavizantes proporcionará información sobre el algoritmo usado para la aproximación.

-Programación de Splines Cúbicos Suavizantes en Matlab, redirecciona al usuario a un documento en formado .pdf donde se detalla el código Matlab o M usado en dicho programa.

-Cómo utilizar esta interfaz, redireccionará al usuario a esta guía, donde se explica el funcionamiento de la interfaz Gráfica.

6.5 La opción ‘‘Acerca de’’.

Al seleccionar esta opción, el usuario visualizará en su pantalla un cuadro de texto donde se explica el propósito del presente programa, así como los nombres de los creadores.

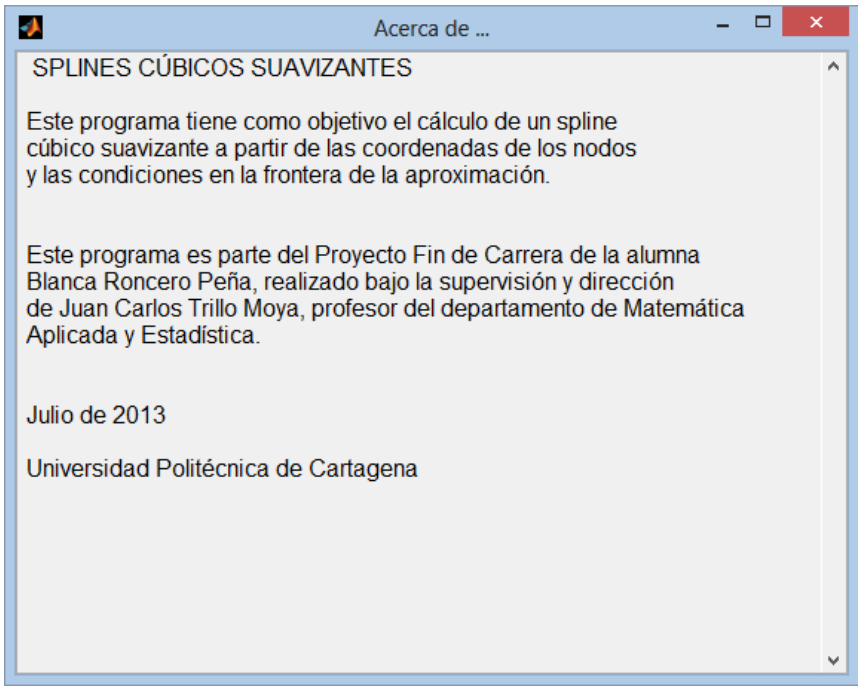


Figura 6.10: Acerca de.

6.6 Ejemplo de introducción de datos.

Vamos a ilustrar un ejemplo de cómo se ejecuta la interfaz creada. Para ello el usuario introducirá en la ventana de comandos del programa (“Command Window”) el siguiente comando. >> SplinesSuavizantes

A continuación, se mostrará la siguiente pantalla, donde pulsaremos el botón ‘’Comenzar’’, tal y como se puede ver en la siguiente figura:

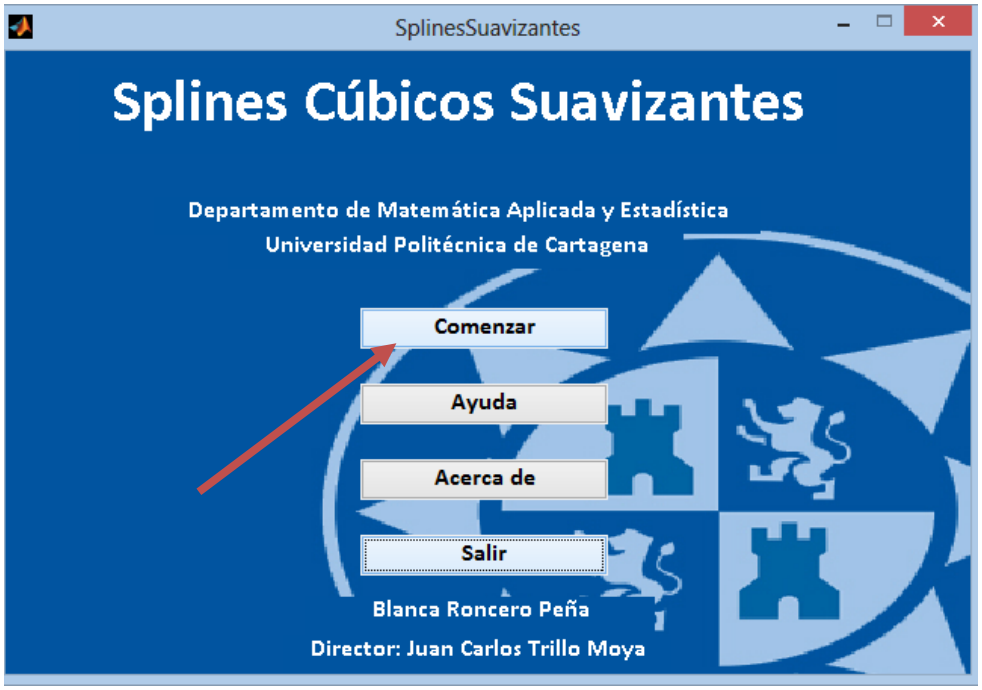


Figura 6.11: Menú principal. Selección de ‘’Comenzar’’

A partir de este punto es cuando debemos introducir los datos necesarios para poder realizar la aproximación mediante splines cúbicos suavizantes, para ello pinchamos en el botón cargar en el cuadro de datos iniciales, tanto para nodos como para coeficientes de peso. Los valores introducidos se muestran a continuación.

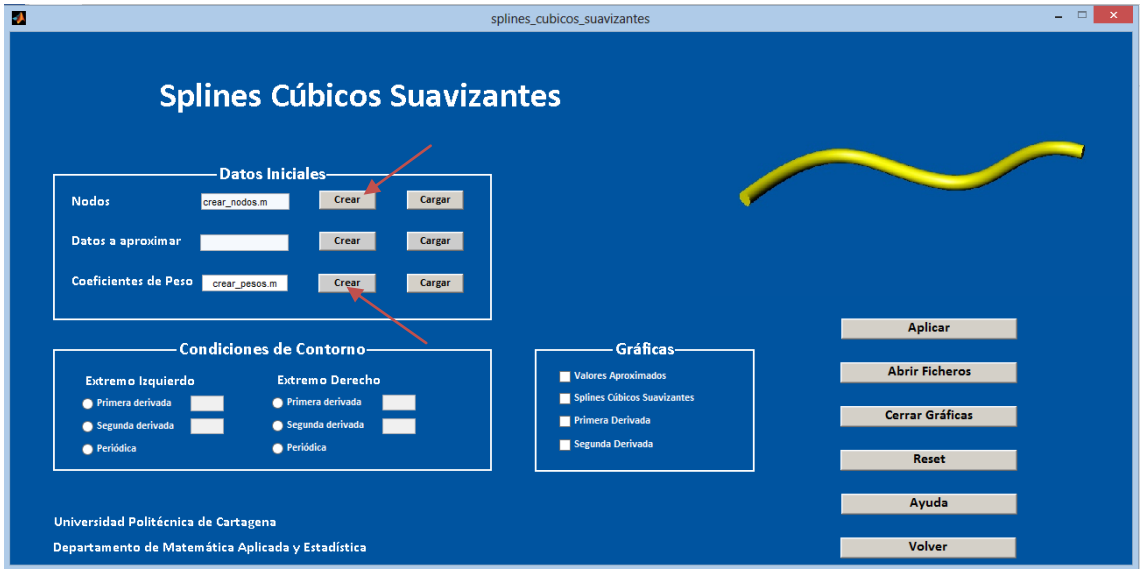


Figura 6.12: Acción, Crear los scripts que definen los nodos y los coeficientes de peso.

Valores de los nodos:

```
function [t,y]=crear_nodos()


% completar vector de abscisas
t=[1,2,3,4,5];

% completar vector de ordenadas
y=[2,-1,0,4,2];
```

Valores de los pesos:

```
function p=crear_pesos()


% completar vector de pesos 0.001*ones(1,3)
p=[0,0.001,0.1,0.1,0];
```

Una vez introducidos los nodos y los pesos, volvemos a la ventana principal de la interfaz gráfica. El siguiente paso es introducir las condiciones de contorno, que en este caso se han elegido periódicas, con lo que pulsando en condiciones periódicas en el extremo izquierdo, automáticamente se marcarán en el extremo derecho.

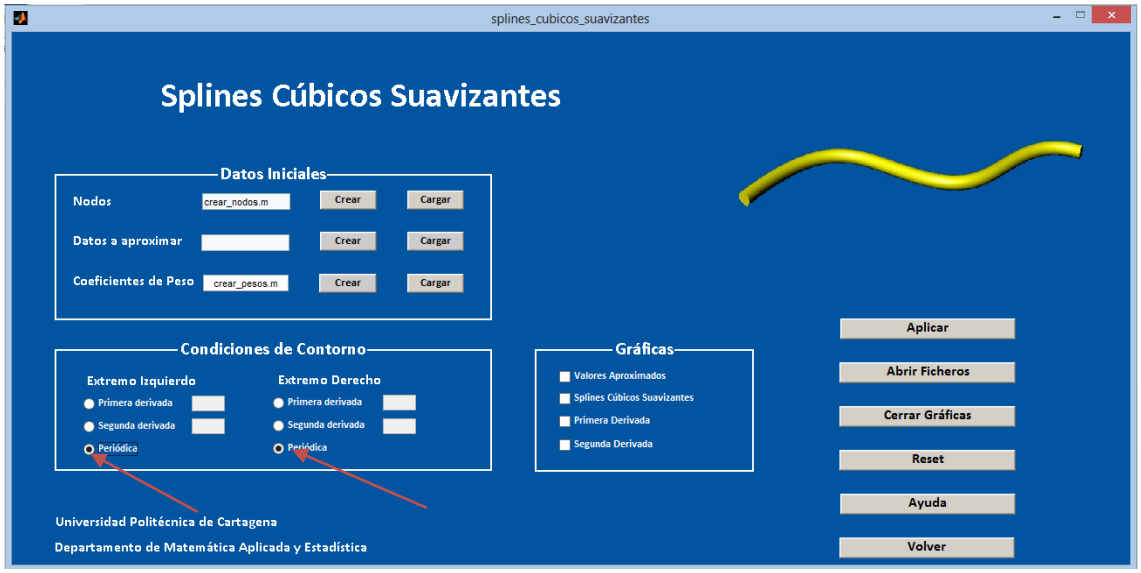


Figura 6.13: Selección de las condiciones de contorno, Periódicas.

Llegados a este punto ya se podría ejecutar la función.

6.7 Ejemplo de salida de datos.

En este primer ejemplo, hemos seleccionado que sólo se muestre la función spline cúbico suavizante, por lo que al pulsar el botón aplicar se mostrará la siguiente pantalla:

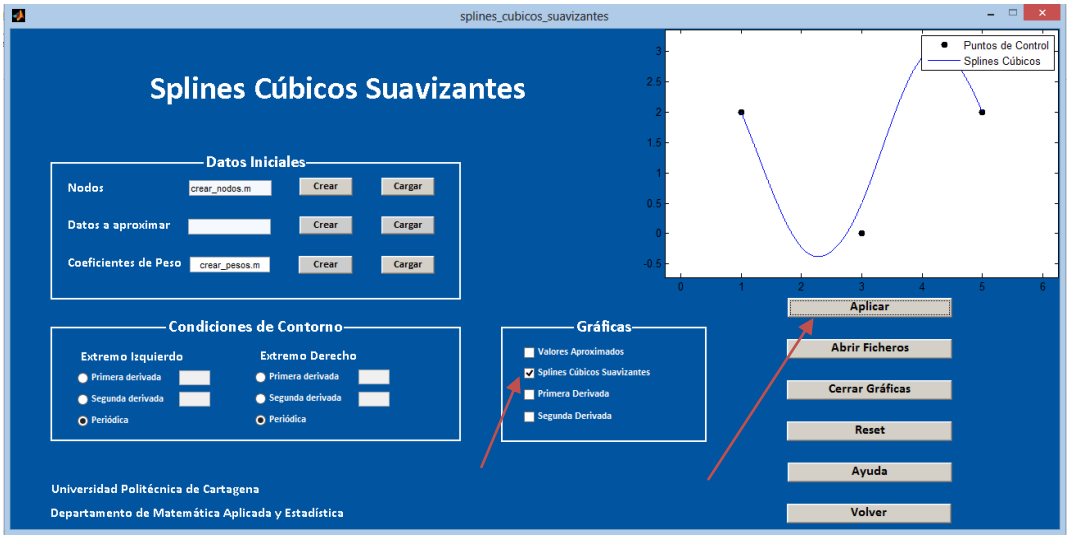


Figura 6.14: Selección de la gráfica a mostrar y aplicación.

Si el usuario decide que también necesita conocer la primera derivada del spline, tan solo tiene que marcar la casilla ‘’Primera Derivada’’ en el panel gráficas, apareciendo una ventana emergente tal y como se muestra a continuación.

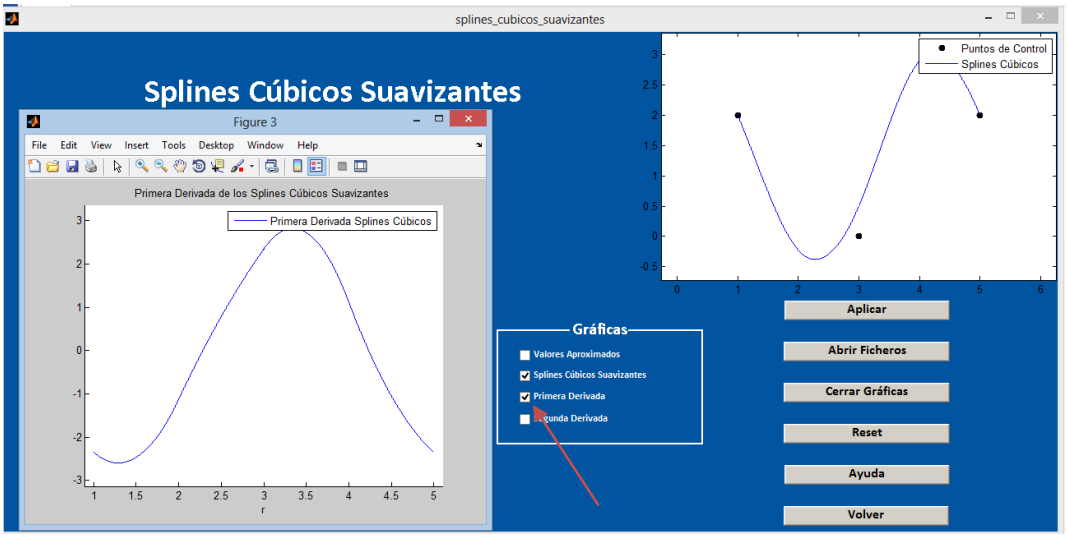


Figura 6.15: Mostrar la primera derivada de la función.

Al pulsar sobre el botón abrir ficheros, se tendrá acceso a un archivo .txt que muestra la función spline cúbico suavizante

```
  + 0.165899 x^3 + 0.502304 x^2 - 5.246544 x + 6.350230
+ 1.451613 x^3 + 15.059908 x^2 - 48.919355 x + 50.023041
 + 0.559908 x^3 + 9.078341 x^2 + 47.633641 x + 78.714286
```

En caso de querer obtener los valores de los datos a aproximar con el spline construido, entonces el procedimiento será análogo al que acabamos de explicar con los nodos. Pero en este caso usaremos la casilla ‘‘Crear’’, correspondiente a la casilla ‘‘Datos aproximar’’. El archivo que se abre en el editor de Matlab al pulsar el botón “Crear” correspondiente es “ crear_datos.m” y el diágolo será tal que:

```
function x=crear_datos()

% completar las abscisas donde se quiere evaluar la reconstrucción
x=[1:0.1:1.5];
```

Esta secuencia queda ilustrada en la siguiente figura:

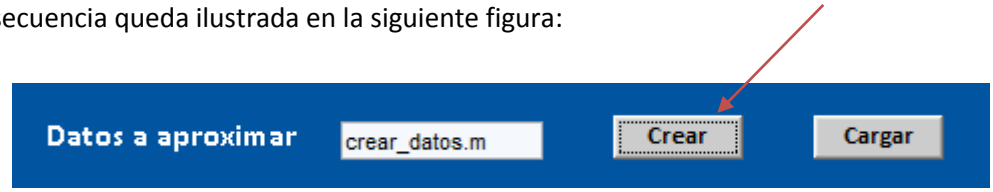


Figura 6.16: Crear datos a aproximar.

Considerando al igual que en el caso anterior una función periódica, procedemos a pulsar de nuevo el botón aplicar. El resultado obtenido puede observarse en la siguiente figura, donde es fácil apreciar que ha aparecido una nueva gráfica con los datos aproximados en el dominio elegido.

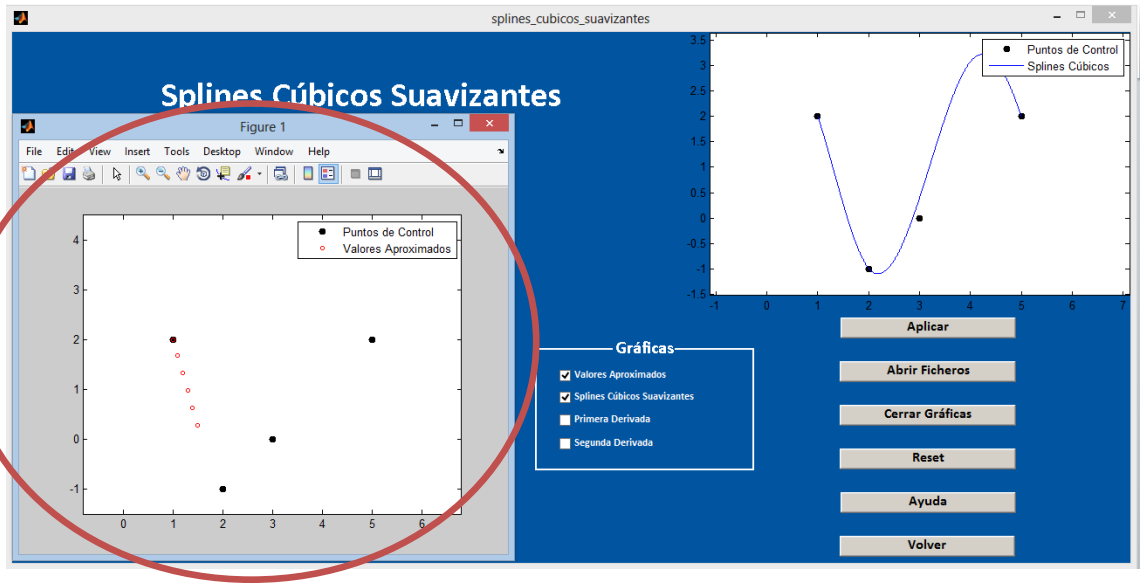


Figura 6.17: Apariencia de la interfaz tras pulsar aplicar.

Al pulsar sobre el botón ‘‘Abrir Ficheros’’, accederemos a un archivo .txt, dónde podremos obtener las coordenadas de los siguientes nodos:

```
1.000000 2.000000
1.100000 1.680886
1.200000 1.338812
1.300000 0.984107
1.400000 0.627099
1.500000 0.278115
```

Por último para salir del programa, pulsando el botón volver, seremos devueltos al menú principal, donde pulsando el botón salir, el programa por seguridad nos preguntará si queremos salir, al pulsar Sí, salimos del programa.

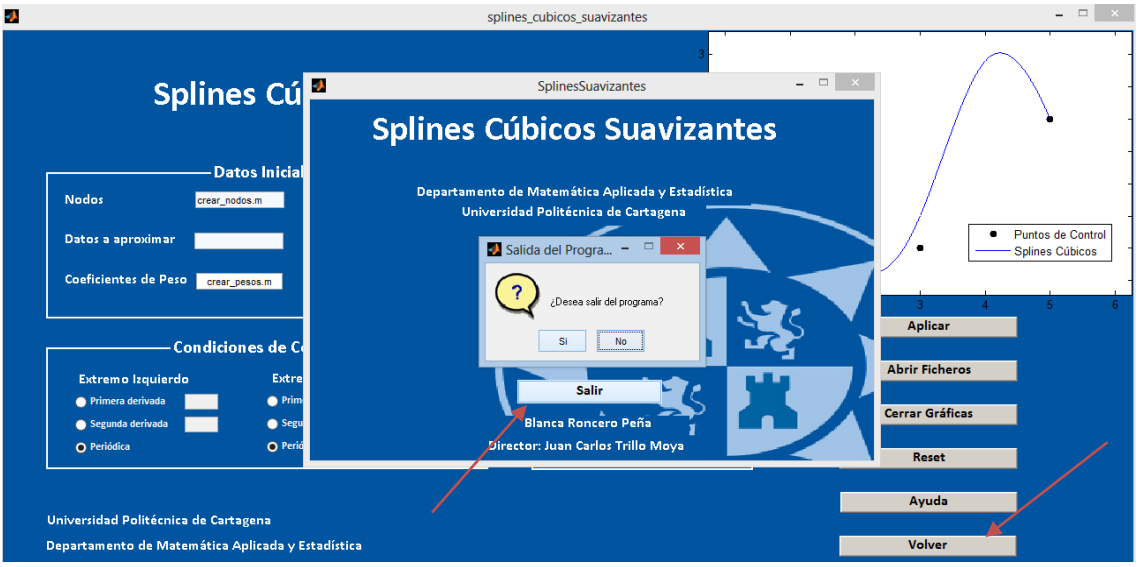


Figura 6.18: Cómo salir del programa.

Capítulo VII
Análisis de un Caso Práctico

Capítulo VII

Análisis de un Caso Práctico

En este capítulo se va a ejecutar en un caso particular de ingeniería naval el algoritmo de cálculo de splines cúbicos suavizantes explicado en los capítulos anteriores mediante la interfaz gráfica.

Para la aplicación se ha elegido un diseño de una embarcación donde aplicaremos lo aprendido sobre splines cúbicos suavizantes para obtener los polinomios de tercer grado que definirán las curvas que componen el plano de formas de la embarcación elegida.

Para obtener las expresiones que definen las curvas suaves, lo primero será obtener los nodos o puntos donde se va a realizar la aproximación.

7.1 Obtención de los datos de entrada.

Para obtener los datos de entrada, es decir las coordenadas de los puntos o nodos donde vamos a realizar la aproximación mediante splines, se podrá optar por usar alguno de los programas de dibujo asistido como Rhinoceros, Solidworks, Autocad,....

En este caso hemos optado por usar Rhinoceros. En dicho programa fue realizado anteriormente el plano de formas original con el cual vamos a obtener los datos necesarios para realizar la aproximación. Los datos pertenecen a una carena de un buque con eslora entre perpendiculares, $L_{pp,}$ de 53,7 metros, la cual ya fue alisada previamente.

Puesto que en una aproximación mediante splines suavizantes no se exige que la curva pase necesariamente por los puntos, vamos a realizar dos estudios prácticos:

1. Introduciendo un cierto ruido a las mediciones.
2. Medición exacta.

Estos dos tipos de mediciones representan dos realidades. La primera de ellas corresponde a mediciones que pueden ser obtenidas de forma experimental, las cuales es imposible que no contengan ningún error. El segundo tipo de mediciones, podría representar unos puntos obtenidos exactamente de una cartilla de trazado o incluso con un programa de diseño asistido por ordenador como puede ser Rhinoceros. En caso de obtener una medición exacta, el algoritmo seguiría siendo válido, ya que si se cumple que $\rho_i = 0$ para todo i, el spline suavizador resulta ser un spline interpolante.

Los splines cúbicos suavizantes también nos permitirán obtener la curva en caso de utilizar una carena sin alisar, obteniéndose muy buenos resultados.

Para conocer las coordenadas de los puntos o nodos de aproximación podremos obtenerlos directamente de la cartilla de trazado, o del plano de formas.

En este caso hemos optado por obtener los puntos manualmente del plano de formas del buque, para como ya se ha comentado, usaremos el programa Rhinoceros. Obtendremos los nodos y a continuación introduciremos los datos en la interfaz gráfica desarrollada en Matlab.

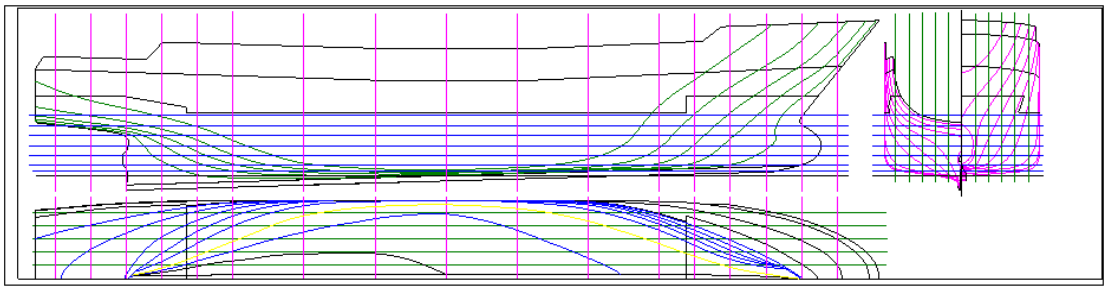


Figura 7.1: Plano de formas y línea de agua seleccionada.

La curva seleccionada ha sido una línea de agua, exactamente la línea de agua 2 representada en la figura 7.1 en color amarillo.
Los valores exactos de los nodos donde se va a realizar la aproximación son:

Abcisas:[-8.45192,-12.3096,-15.4722,-18.284,-21.4086,-25.8184,-33.2787,-34.8182,-42.521,-46.411,-48.8637,-51.3927,-54.0604,-57.0541]
Ordenadas:[0.609,1,2,3,4,5,5.54861,5.54861,5,4,3,2,1,0.609]

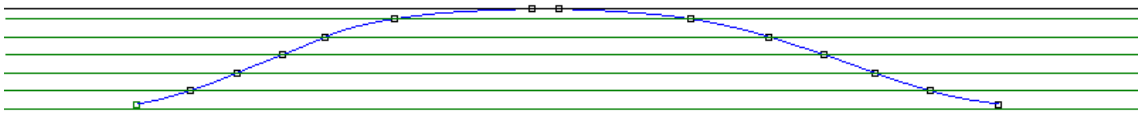


Figura 7.2: Puntos obtenidos de la curva.

En general se elegirá un número pequeño de puntos ya que cuanto menor número de nodos tengamos mejor será el alisado de esta forma se tendrá menor número de cambios en su curvatura, sin embargo el número de puntos deberá ser suficiente para que el trazado de la curva sea correcto. En este caso las formas del buque ya han sido alisadas, por lo tanto no habrá cambios bruscos y hemos elegido 14 puntos.

7.2 Introducción de los datos de entrada en la interfaz.

Vamos a analizar los dos posibles casos ya descritos anteriormente.

7.2.1 Caso 1: coordenadas de los nodos medidas con exactitud.

Ejecutamos la interfaz gráfica como ya hemos explicado en el capítulo anterior. Introducimos las coordenadas de los nodos que se van a utilizar para obtener la curva suave. Para ello utilizamos el valor de las mediciones exactas realizadas anteriormente.

Los valores de los nodos introducidos han sido los que se muestran a continuación en el archivo crear_nodos.m. Es muy importante que los nodos estén ordenados de menor a mayor.

```
function [t,y]=crear_nodos()

% completar vector de abscisas
t=[-57.0541,-54.0604,-51.3927,-48.8637,-46.411,-42.521,-34.8182,-
33.2787,-25.8184,-21.4086,-18.284,-15.4722,-12.3096,-8.45192];
% completar vector de ordenadas
y=[0.609,1,2,3,4,5,5.54861,5.54861,5,4,3,2,1,0.609];
```

A continuación introducimos los valores de los pesos o coeficientes de suavizado. En este caso al ser mediciones exactas pondremos el valor 0 para cada coeficiente de peso en el archivo crear_pesos.m:

```
function p=crear_pesos()

% completar vector de pesos
p=[0,0,0,0,0,0,0,0,0,0,0,0,0,0];
```

Como condiciones de contorno elegimos periódicas, ya que en este caso es aplicable al tener los dos extremos las mismas coordenadas en el eje de ordenadas.

En cuanto a gráficas, seleccionamos que el programa nos muestre la función spline, la primera y segunda derivada, para así poder realizar las comprobaciones necesarias acerca de la suavidad de la curva.

La apariencia que tendrá la interfaz gráfica al introducir toda esta información será la que se muestra en la siguiente imagen:

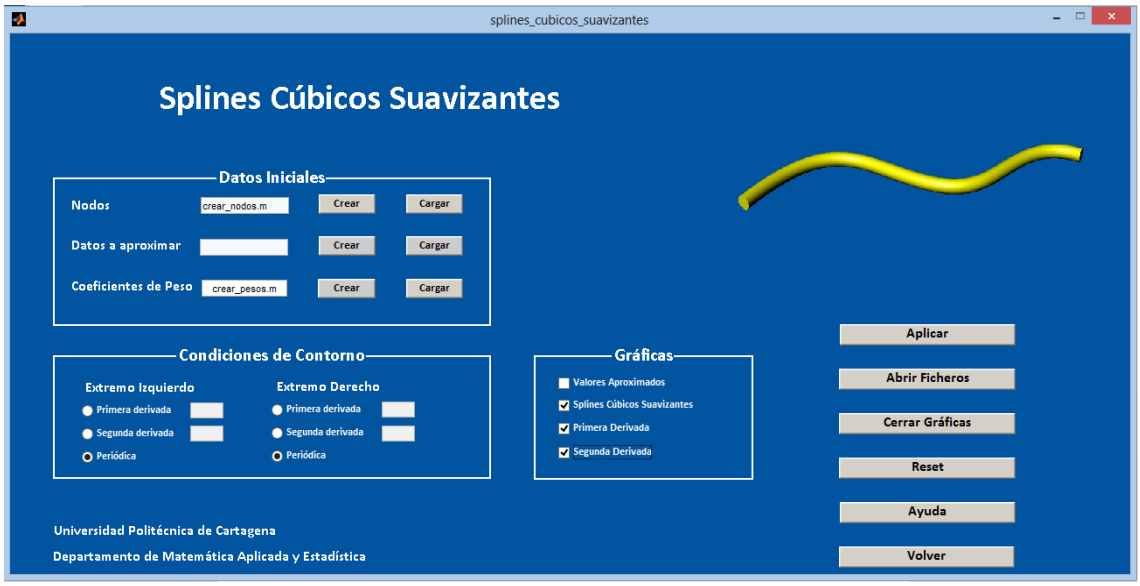


Figura 7.3: Apariencia de la interfaz al introducir los datos.

Pulsando el botón “Aplicar” aparecerán en pantalla las siguientes gráficas:

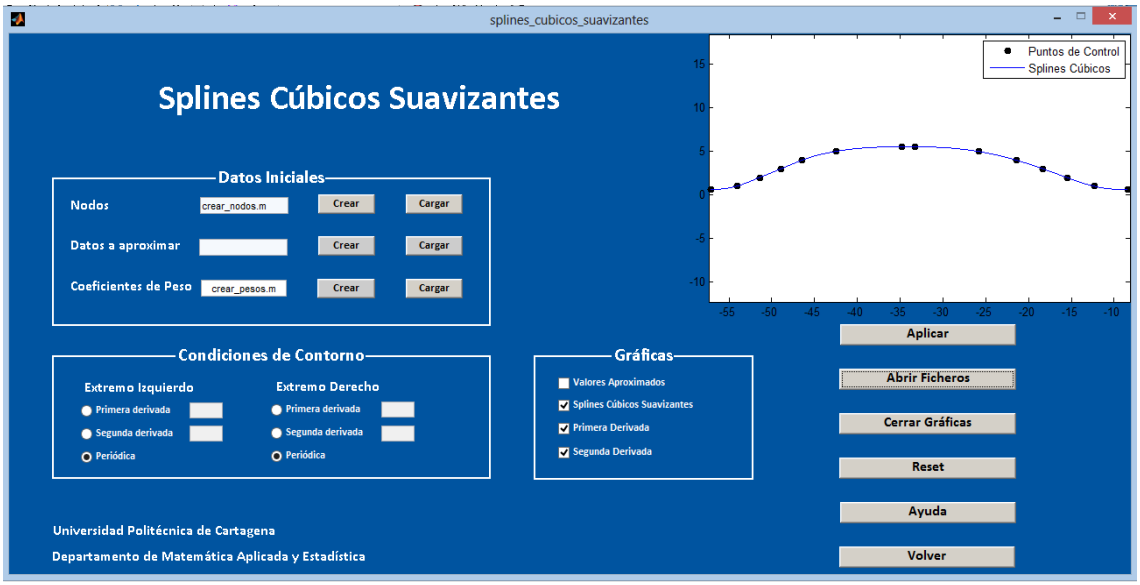


Figura 7.4: Pantalla principal tras pulsar aplicar.

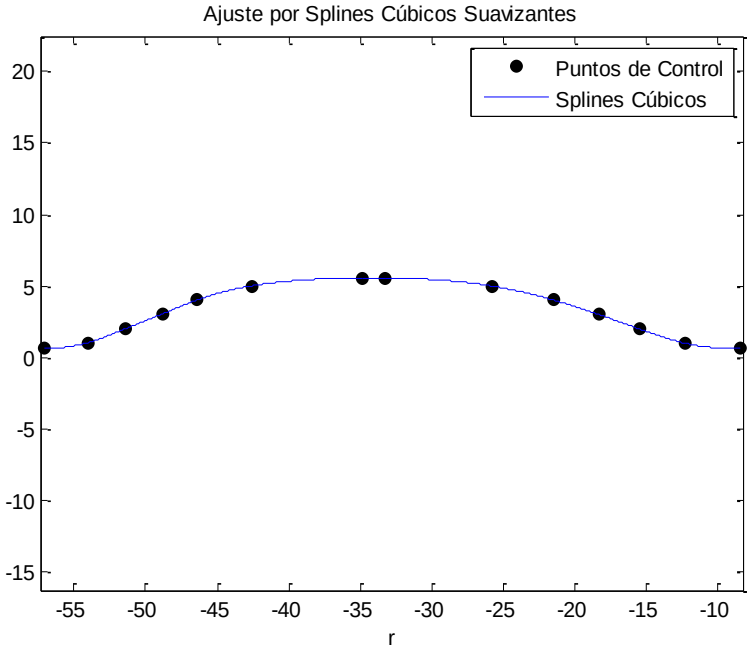


Figura 7.5: Gráfica del spline.

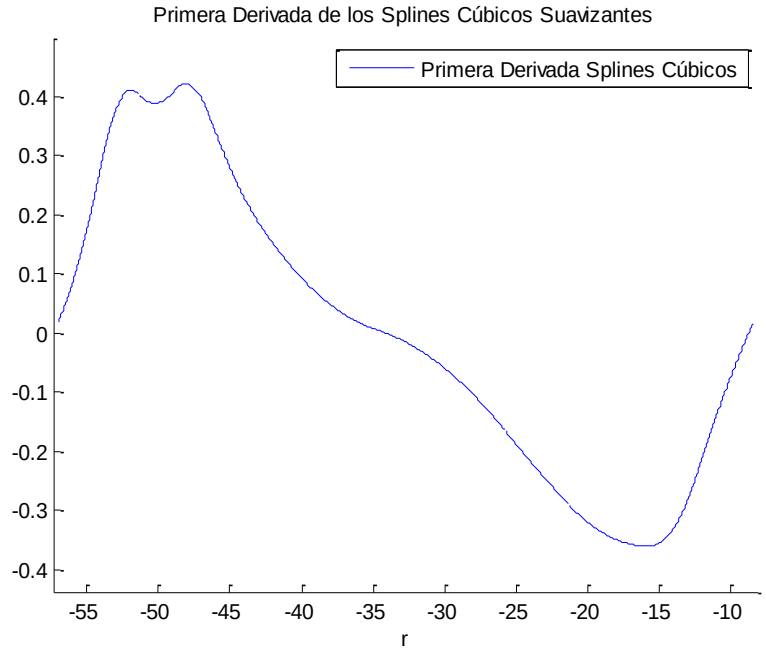


Figura 7.6: Gráfica de la primera derivada del spline.

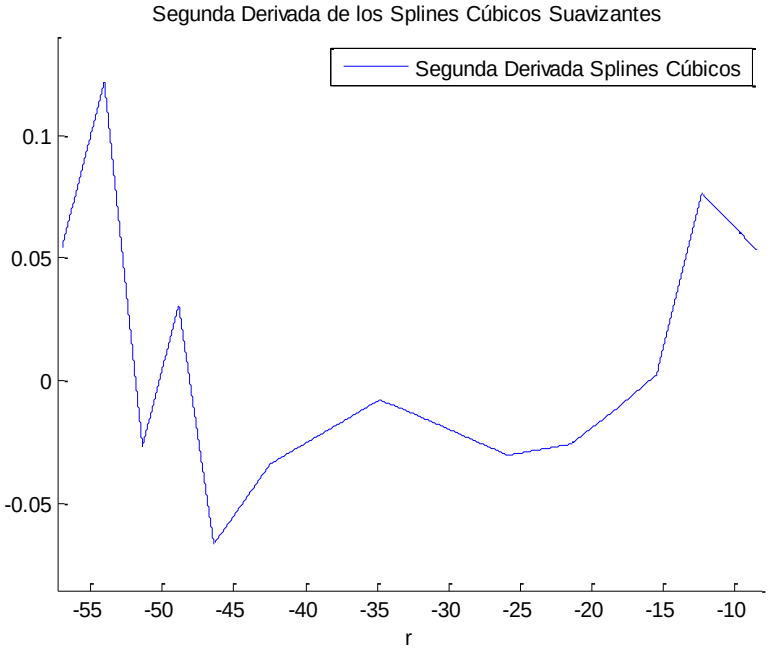


Figura 7.7: Gráfica de la segunda derivada del spline.

Por último pulsamos el botón “Abrir Ficheros” y copiamos las expresiones de los polinomios:

```
 + 0.003796 x^3 + 0.676420 x^2 + 40.133425 x + 793.488061
+ 0.009272 x^3 + 1.442953 x^2 + 74.440717 x + 1271.153259
 + 0.003784 x^3 + 0.569986 x^2 + 29.009645 x + 501.044538
 + 0.006610 x^3 + 0.953689 x^2 + 45.442747 x + 711.628578
  + 0.001387 x^3 + 0.159777 x^2 + 6.234323 x + 87.832934
  + 0.000571 x^3 + 0.055704 x^2 + 1.809000 x + 25.109880
  + 0.000362 x^3 + 0.041747 x^2 + 1.584060 x + 14.270210
  + 0.000428 x^3 + 0.048364 x^2 + 1.804246 x + 16.712704
   + 0.000174 x^3 + 0.001749 x^2 + 0.600739 x + 6.355168
   + 0.000750 x^3 + 0.035272 x^2 + 0.191844 x + 0.699133
   + 0.000832 x^3 + 0.039774 x^2 + 0.274157 x + 0.197461
  + 0.003916 x^3 + 0.182931 x^2 + 2.489109 x + 11.225928
   + 0.001001 x^3 + 0.001325 x^2 + 0.253613 x + 2.053241
```

7.2.2 Caso 2: coordenadas con un cierto grado de error.

En este caso vamos a introducir en dos de los puntos un error grande, para así poder visualizar con facilidad la ventaja de un spline suavizante. Para ello seguiremos el mismo procedimiento del caso anterior.

Los valores de los nodos introducidos han sido los que se muestran a continuación en el archivo crear_nodos.m. Es muy importante que los nodos estén ordenados de menor a mayor.

```
function [t,y]=crear_nodos()

% completar vector de abscisas
t=[-57.0541,-54.0604,-51.3927,-48.8637,-46.411,-42.521,-34.8182,-
33.2787,-25.8184,-21.4086,-18.284,-15.4722,-12.3096,-8.45192];
% completar vector de ordenadas
y=[0.609,1,2,3,4,6,5.54861,5.54861,6,4,3,2,1,0.609];
```

Como se puede observar, los nodos 6 y 9 contienen un error bastante grande. A continuación. Introducimos los pesos, ya que el error es exagerado en los nodos 6 y 9, introduciremos los pesos correspondientes con un valor alto como se puede ver a continuación. En la práctica no será necesario introducir un valor tan elevado de dicho parámetro.

```
function p=crear_pesos()


% completar vector de pesos
p=[0,0,0,0,0,100,0,0,100,0,0,0,0,0];
```

A continuación, volvemos a marcar condiciones periódicas, y pedimos que nos muestre las gráficas del spline, primera y segunda derivada como en el caso anterior. Pulsamos aplicar, y el resultado obtenido es el siguiente:

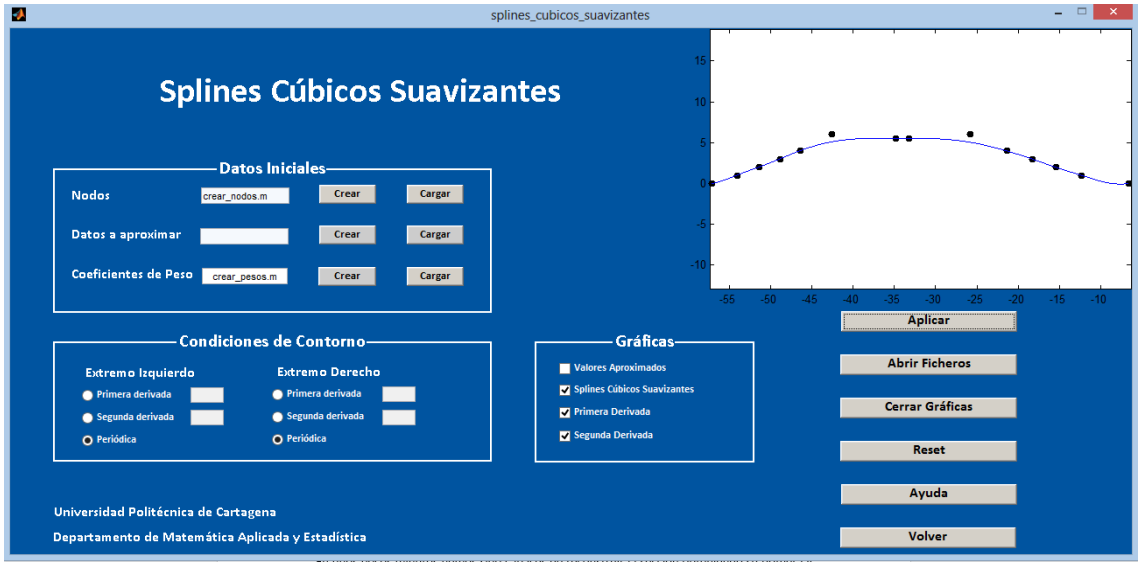


Figura 7.8: Apariencia de la interfaz al pulsar aplicar.

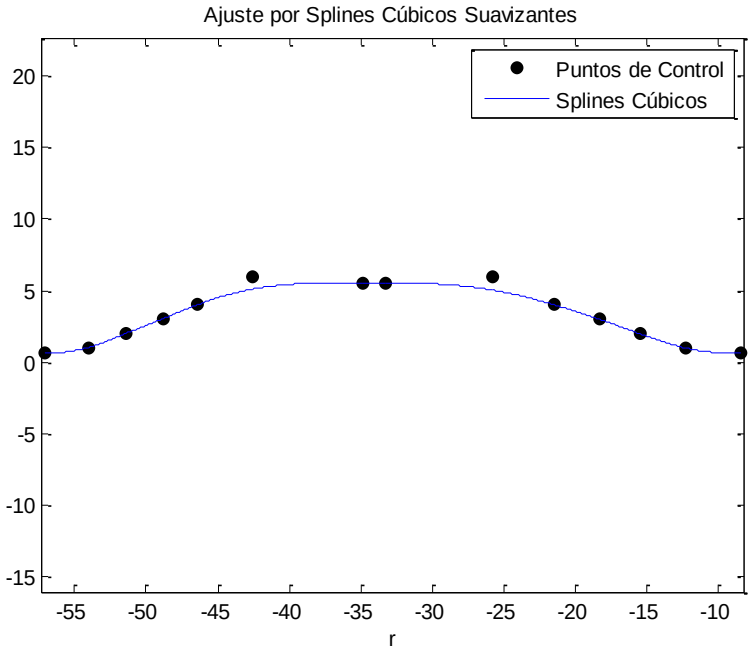


Figura 7.9: Gráfica de la función spline.

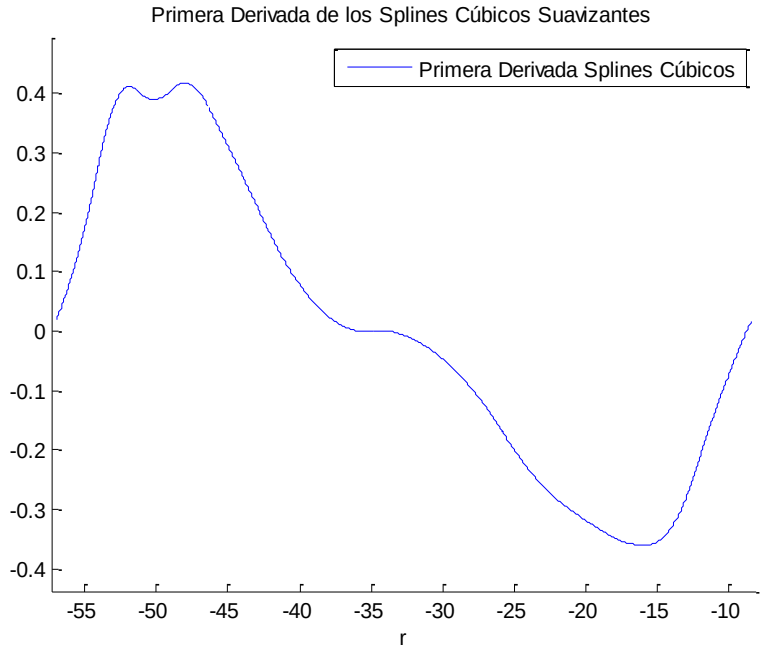


Figura 7.10: Gráfica de la primera derivada del spline.

Figura 7.11: Gráfica de la segunda derivada del spline

Y los polinomios obtenidos fueron:

```
 + 0.003770 x^3 + 0.672066 x^2 + 39.888467 x + 788.896677
+ 0.009168 x^3 + 1.426270 x^2 + 73.548391 x + 1255.250639
 + 0.003363 x^3 + 0.505766 x^2 + 25.744145 x + 445.719870
 + 0.004960 x^3 + 0.714378 x^2 + 33.876611 x + 525.377045
  + 0.000189 x^3 + 0.050074 x^2 + 3.045584 x + 48.410772
  + 0.001197 x^3 + 0.126782 x^2 + 4.474515 x + 58.176597
  + 0.001053 x^3 + 0.108299 x^2 + 3.710603 x + 36.820426
  + 0.000723 x^3 + 0.075314 x^2 + 2.612898 x + 24.643692
   + 0.000787 x^3 + 0.041625 x^2 + 0.406277 x + 1.339734
   + 0.000204 x^3 + 0.004204 x^2 + 0.394861 x + 4.377352
   + 0.000999 x^3 + 0.047788 x^2 + 0.402038 x + 0.479485
  + 0.003878 x^3 + 0.181428 x^2 + 2.469738 x + 11.143438
   + 0.000989 x^3 + 0.001694 x^2 + 0.257290 x + 2.065321
```

7.3 Comparación entre la interpolación y la aproximación.

Para simplificar, primero vamos a demostrar la exactitud de los splines cúbicos suavizantes cuando las coordenadas de los nodos medidos son exactas. Para comprobar que la curva es correcta tratamos la imagen de la gráfica obtenida con Matlab. Convertimos el fondo en transparente y superponemos en Rhinoceros ambas imágenes. El resultado se puede ver en la siguiente figura:

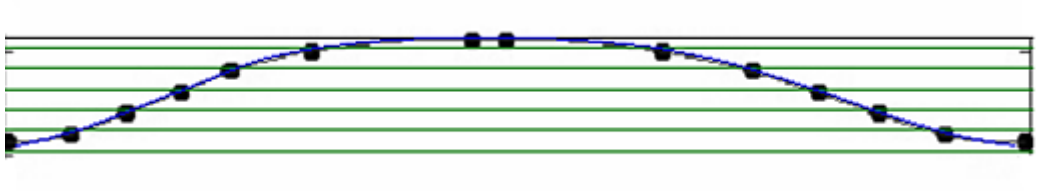


Figura 7.12: Superposición del Spline y la curva real.

Como podemos observar, la curva reconstruida y la curva original son prácticamente coincidentes en todos sus puntos.

Después de ejecutar los dos casos prácticos anteriores, uno donde las mediciones eran exactas, y un segundo caso donde se introdujo a las mediciones un grado de ruido elevado en dos de sus nodos, se ha decidido superponer dichas gráficas de la función spline, para así poder comparar ambas y verificar la eficacia de la función spline cúbico suavizante, véase figura 7.13.

En la siguiente gráfica 7.13 se muestra en color rojo, la gráfica de la función spline cúbico suavizante para un coeficiente de peso con valor 0, es decir un spline cúbico interpolante. En azul se ha mostrado la función spline cuando las mediciones contienen ciertas desviaciones, para un coeficiente de peso igual a 100.

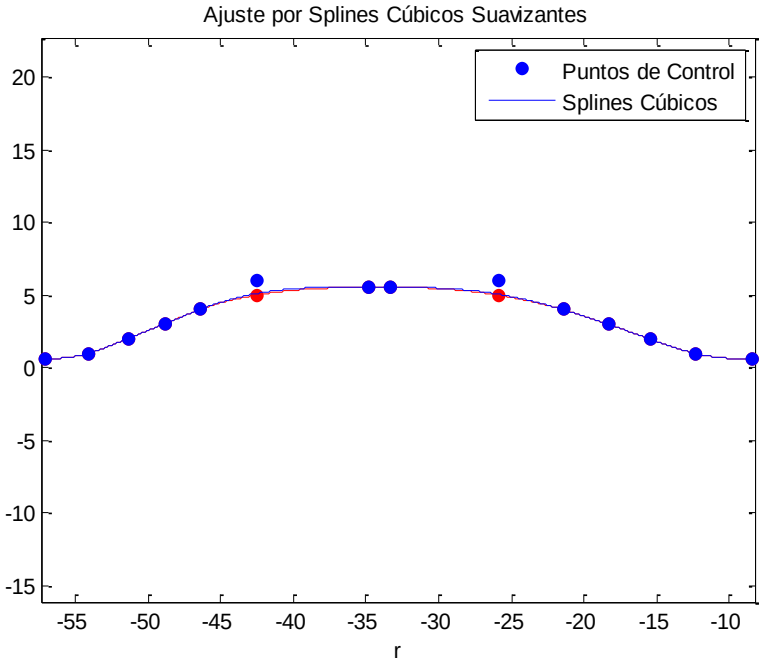


Figura 7.13: Comparación entre la función spline con coordenadas exactas (rojo), y con coordenadas desviadas (azul).

Como se puede comprobar, las gráficas son prácticamente coincidentes, lo cual demuestra que si en la práctica cometiéramos un cierto error a la hora de medir las coordenadas de los nodos que componen la superficie de nuestro casco, teniendo una ligera idea del error que se ha cometido, es posible obtener la función aproximante que se adapta perfectamente a la geometría del casco.

Si en lugar de realizar una aproximación, se hubiera realizado una interpolación, el resultado hubiera sido muy lejano al objetivo que se pretende alcanzar. Esto se ilustra en la siguiente figura, donde en color verde se muestra el resultado al interpolar entre los nodos que contienen cierto grado de ruido, en rojo la aproximación por splines cúbicos suavizantes y en azul se muestra la curva original buscada.

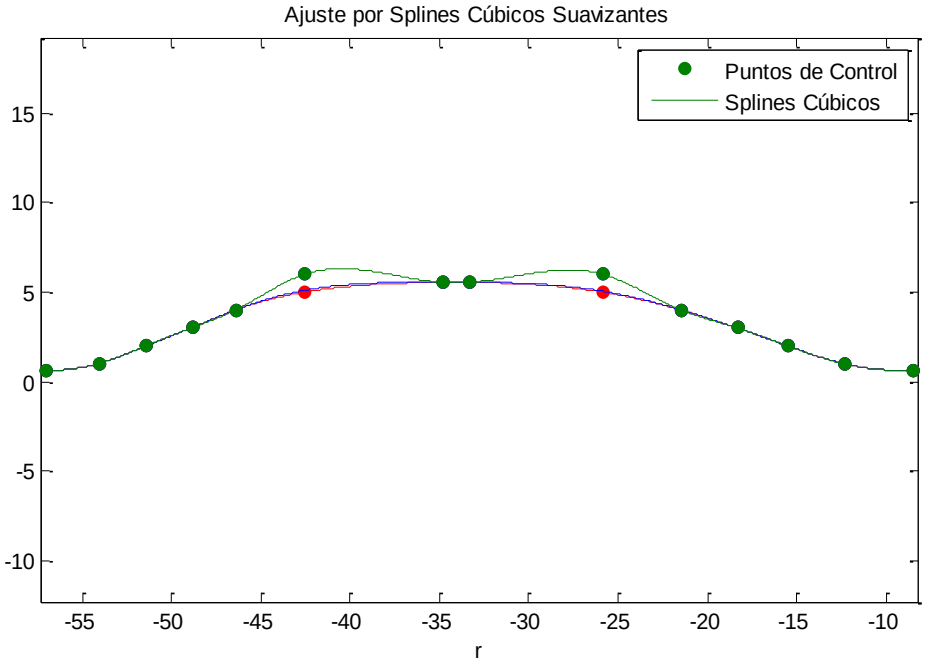


Figura 7.14: Comparación entre la aproximación (rojo) y la interpolación (verde). El gráfico azul es el objetivo a lograr.

Con las ilustraciones anteriores, queda demostrada la utilidad de una aproximación, y la potencia que presentan las curvas splines cúbicos suavizantes.

Conclusiones

Conclusión

A lo largo de este Proyecto Fin de Carrera se ha presentado una visión global de las diferentes herramientas que existen para el diseño de curvas suaves en el diseño naval. Se ha realizado un pequeño estudio de todos los métodos usados en la actualidad tanto desde un punto de vista matemático como desde un punto de vista práctico e interactivo, constatando que en la actualidad la mayoría de programas de dibujo tanto industrial como naval basan su funcionamiento en el uso de curvas spline.

Después de desarrollar el algoritmo de cálculo de los splines cúbicos suavizantes se ha observado que estos dan muy buenos resultados a la hora de definir la compleja geometría del casco de una embarcación, realizando una aproximación que se ajusta a la realidad a pesar de que las mediciones contengan cierto grado de ruido. Este tipo de ajuste no podría realizarse con otros métodos numéricos como puede ser la interpolación segmentaria de Lagrange o los splines cúbicos interpolantes. Asimismo este algoritmo presenta la ventaja de poder ser adaptado a la función spline cúbico interpolante siempre que las mediciones realizadas fueran exactas, con el simple hecho de indicarlo los coeficientes de suavizado.

La aplicación de los splines cúbicos suavizantes en este estudio se ha centrado en el campo del diseño naval. Sin embargo esta técnica tiene múltiples aplicaciones tanto industriales, estadísticas como científicas. En el campo de la estadística, por ejemplo, estas funciones son usadas a la hora de realizar predicciones mediante modelos de regresión no paramétrica, obteniéndose muy buenos resultados debido a la alta flexibilidad que éstos presentan gracias al coeficiente de suavizado o de peso.

Durante la realización de este PFC se han empleado otros programas, como Maple. Maple es ampliamente utilizado en el mundo de la ingeniería y de las matemáticas por ser una herramienta muy potente para el cálculo simbólico. Maple nos ha sido útil a la hora de buscar las expresiones de las matrices de coeficientes necesarias para desarrollar el algoritmo de cálculo de la función spline.

A lo largo de este proyecto he adquirido una amplia práctica en programación, en matemáticas y en especial en la construcción de curvas suaves, lo cual es una ventaja importante para una persona que quiera dedicar su vida profesional a la arquitectura naval.

Anexos

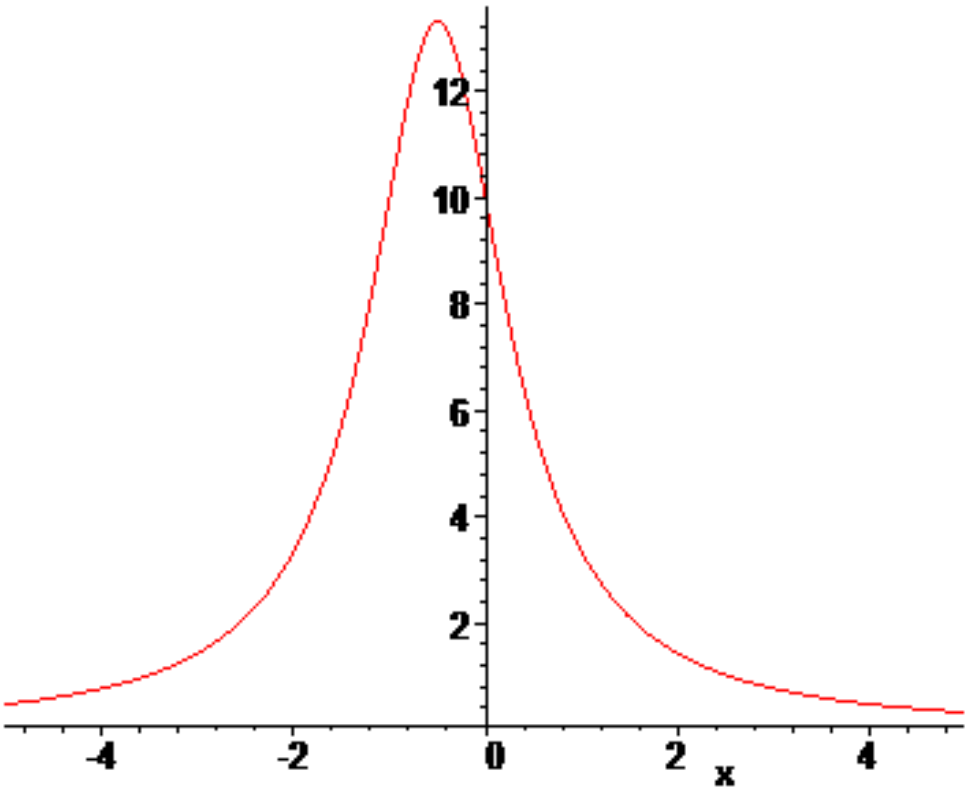
Anexo I. Función de interpolación de Lagrange mediante MAPLE.

Anexo II. Código Matlab para la interfaz gráfica.

Anexo III. Plano de formas

Anexo I. Función de interpolación de Lagrange mediante MAPLE.

```
> creation_poly:=proc(d1,d2,x1,x2,y1,y2)
local x,h:
h:=x2-x1:
unapply(y1*(x2-x)/h + y2*(x-x1)/h
-h*h/6*d1*((x2-x)/h-((x2-x)/h)^3)
-h*h/6*d2*((x-x1)/h-((x-x1)/h)^3),x)
end:
> with(LinearAlgebra):
>
> s:=proc(x::list(numeric),y::list(numeric))
local n,i,j,mat,res,sol,draw,h1,h2,pol:
if nops(x)<>nops(y) then ERROR ("numberofx and y most be
equal") fi:
n:=nops(x):
mat:=[1,seq(0,j=1..n-1)],[seq(0,j=1..n-1),1]:
res:=0,0:
for i from 2 to n-1 do
h1:=x[i]-x[i-1]:
h2:=x[i+1]-x[i]:
mat:=[seq(0,j=1..i-
2),h1*h1,2*(h1*h1+h2*h2),h2*h2,seq(0,j=1..n-i-1)],mat:
res:=6*(y[i+1]-2*y[i]+y[i-1]),res:
od:
sol:=linsolve([mat],[res]):
draw:=NULL:
for i to n-1 do
pol:=creation_poly(sol[i],sol[i+1],x[i],x[i+1],y[i],y[i+1])
:
draw:=plot(pol(z),z=x[i]..x[i+1]),draw:
 od:
eval(draw):
end:
> test1:=s([0,1/4,1/2,3/4,1],[0,1/16,1/4,9/16,1]):
> display(test1,plot(10/(x^2+x+1),x=-5..5,color=red)):
> plot(10/(x^2+x+1),x=-5..5);
f:=x->10/(x^2+x+1);
x1:=[-5,-4,-3,-2,-1,-0.5,-0.25,0,0.25,1,2,3,4,5,6];
fx1:=map(f,x1);
plot([10/(x^2+x+1),spline(x1,fx1,x,'linear')],x=-
5..5,color=[red,blue],style=[line,line]);
plot([10/(x^2+x+1),spline(x1,fx1,x,'cubic')],x=-
5..5,color=[red,blue],style=[line,line]);
```



$$f := x \to 10\,\frac{1}{x^2 + x + 1}$$

$$x1 := [\,-5, -4, -3, -2, -1, -.5, -.25, 0, .25, 1, 2, 3, 4, 5, 6\,]$$

$$fx1 := \left[\frac{10}{21}, \frac{10}{13}, \frac{10}{7}, \frac{10}{3}, 10, 13.33333333, 12.30769231, 10, 7.619047619, \frac{10}{3}, \frac{10}{7}, \frac{10}{13}, \frac{10}{21}, \frac{10}{31}, \frac{10}{43}\right]$$

Anexo II. Código Matlab para la interfaz gráfica.

Código Matlab del Menú Principal.

```
function varargout = principal(varargin)
% SPLINESSUAVIZANTES M-file for SplinesSuavizantes.fig
%      SPLINESSUAVIZANTES, by itself, creates a new SPLINESSUAVIZANTES
or raises the existing
%      singleton*.
%      H = SPLINESSUAVIZANTES returns the handle to a new
SPLINESSUAVIZANTES or the handle to
%      the existing singleton*.
%
%      SPLINESSUAVIZANTES('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in SPLINESSUAVIZANTES.M with the given
input arguments.
%
%      SPLINESSUAVIZANTES('Property','Value',...) creates a new
SPLINESSUAVIZANTES or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before SplinesSuavizantes_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to SplinesSuavizantes_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
SplinesSuavizantes

% Last Modified by GUIDE v2.5 27-Mar-2013 15:22:02

% Begin initialization code - DO NOT EDIT

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @SplinesSuavizantes_OpeningFcn,
...
                   'gui_OutputFcn',  @SplinesSuavizantes_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before SplinesSuavizantes is made visible.
function SplinesSuavizantes_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to splines_cubicos_suavizantes
(see VARARGIN)

% Choose default command line output for splines_cubicos_suavizantes
handles.output = hObject;


% Update handles structure
guidata(hObject, handles);

% UIWAIT makes splines_cubicos_suavizantes wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%% CENTRAR INTERFAZ GRÁFICA EN PANTALLA %%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

set( handles.output,'Units', 'pixels' );
screenSize = get(0, 'ScreenSize');

position = get( handles.output,'Position');
position(1) = (screenSize(3)-position(3))/2;
position(2) = (screenSize(4)-position(4))/2;
set( handles.output,'Position', position );
%%%Introducimos el fondo con el escudo de la UPCT%%%
% create an axes that spans the whole gui
ah = axes('unit', 'normalized', 'position', [0 0 1 1]);
% import the background image and show it on the axes
bg = imread('fondo_principal.jpg'); imagesc(bg);
% prevent plotting over the background and turn the axis off
set(ah,'handlevisibility','off','visible','off')
% making sure the background is behind all the other uicontrols
uistack(ah, 'bottom');

% Choose default command line output for SplinesSuavizantes
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
% UIWAIT makes SplinesSuavizantes wait for user response (see
UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = SplinesSuavizantes_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
splines_cubicos_suavizantes;


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
menu_ayuda;


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fid = fopen('acerca_de.txt');
str = fread(fid,inf,'*char').';
fclose(fid);
f = figure('menubar','none','NumberTitle','off','Name','Acerca de
...');
h=uicontrol('style','edit','parent',f,'min',0,'max',2,'enable','inacti
ve'...
    ,'units','normalized','position',[0 0 1 1],'string',str,...
    'horizontalalignment','left','fontsize',12);
drawnow

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
salir=questdlg('¿Desea salir del programa?','Salida del
Programa','Si','No','No');
switch salir
    case 'Si'
        close all;
    case 'No'
        return;
end
```

Código Matlab de la Pantalla Principal.

```
function varargout = splines_cubicos_suavizantes(varargin)
% SPLINES_CUBICOS_SUAVIZANTES M-file for
splines_cubicos_suavizantes.fig
%      SPLINES_CUBICOS_SUAVIZANTES, by itself, creates a new
SPLINES_CUBICOS_SUAVIZANTES or raises the existing
%      singleton*.
%      H = SPLINES_CUBICOS_SUAVIZANTES returns the handle to a new
SPLINES_CUBICOS_SUAVIZANTES or the handle to
%      the existing singleton*.

SPLINES_CUBICOS_SUAVIZANTES('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in SPLINES_CUBICOS_SUAVIZANTES.M with
the given input arguments.
%      SPLINES_CUBICOS_SUAVIZANTES('Property','Value',...) creates a
new SPLINES_CUBICOS_SUAVIZANTES or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before
splines_cubicos_suavizantes_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to
splines_cubicos_suavizantes_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help
splines_cubicos_suavizantes
% Last Modified by GUIDE v2.5 13-Apr-2013 18:05:37
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@splines_cubicos_suavizantes_OpeningFcn, ...
                   'gui_OutputFcn',
@splines_cubicos_suavizantes_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before splines_cubicos_suavizantes is made
visible.
function splines_cubicos_suavizantes_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to splines_cubicos_suavizantes
(see VARARGIN)

% Choose default command line output for splines_cubicos_suavizantes
handles.output = hObject;


% Update handles structure
guidata(hObject, handles);

% UIWAIT makes splines_cubicos_suavizantes wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%% CENTRAR INTERFAZ GRÁFICA EN PANTALLA %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

set( handles.output,'Units', 'pixels' );
screenSize = get(0, 'ScreenSize');

position = get( handles.output,'Position');
position(1) = (screenSize(3)-position(3))/2;
position(2) = (screenSize(4)-position(4))/2;
set( handles.output,'Position', position );

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Cuadros de escritura de las condiciones de contorno aparezcan
%deasactivados hasta que se marque alguna opción
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
set(handles.edit_primera_izq,'Enable','off');
set(handles.edit_segunda_izq,'Enable','off');
set(handles.edit_primera_drch,'Enable','off');
set(handles.edit_segunda_drch,'Enable','off');



% --- Outputs from this function are returned to the command line.
function varargout = splines_cubicos_suavizantes_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% INCLUIMOS EL ESCUDO DE LA UPCT %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Importamos imagen *.jpg
grafica_inicial=imread('escudo.jpg');
axes(handles.grafica);

%imshow(escudo_upct);
imagesc(grafica_inicial); axis off


% --- Executes on button press in push_crearnodos.
function push_crearnodos_Callback(hObject, eventdata, handles)
% hObject    handle to push_crearnodos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

edit crear_nodos.m

set(handles.edit_nodos,'String','crear_nodos.m');


% --- Executes on button press in push_creardatos.
function push_creardatos_Callback(hObject, eventdata, handles)
% hObject    handle to push_creardatos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

edit crear_datos.m

set(handles.edit_datos,'String','crear_datos.m');


% --- Executes on button press in crear_pesos.
function crear_pesos_Callback(hObject, eventdata, handles)
% hObject    handle to crear_pesos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

edit crear_pesos.m

set(handles.edit_pesos,'String','crear_pesos.m');


% --- Executes on button press in push_cargarnodos.
function push_cargarnodos_Callback(hObject, eventdata, handles)
% hObject    handle to push_cargarnodos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename,pathname,filterindex]=uigetfile( ...
    { '*.m', 'Archivo que contiene los nodos de reconstrucción
(*.m)'},'Seleccione un archivo');
if filterindex == 1
    set(handles.edit_nodos,'String',filename);
end
```

```
% --- Executes on button press in push_cargardatos.
function push_cargardatos_Callback(hObject, eventdata, handles)
% hObject    handle to push_cargardatos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[filename,pathname,filterindex]=uigetfile( ...
    { '*.m', 'Archivo que contiene los datos donde aproximar
(*.m)'},'Seleccione un archivo');

if filterindex == 1
    set(handles.edit_datos,'String',filename);
end


% --- Executes on button press in cargar_pesos.
function cargar_pesos_Callback(hObject, eventdata, handles)
% hObject    handle to cargar_pesos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


[filename,pathname,filterindex]=uigetfile( ...
    { '*.m', 'Archivo que contiene los pesos (*.m)'},'Seleccione un
archivo');

if filterindex == 1
    set(handles.edit_pesos,'String',filename);
end


% --- Executes on button press in push_aplicar.
function push_aplicar_Callback(hObject, eventdata, handles)
% hObject    handle to push_aplicar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% LEEMOS LOS DATOS INICIALES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Recogemos los nodos y llamamos al correspondiente programa
nodos= get(handles.edit_nodos,'String');
nodos=fliplr(deblank(fliplr(deblank(nodos)))); %Eliminamos huecos en
blanco

% Comprobamos que no está vacío
if isempty(nodos)==1
    uiwait(msgbox('Debe introducir los nodos por los que ha de pasar
el Spline', 'Mensaje de error',...
        'error','modal'))
    return;
end
% Eliminamos la extensión del archivo para poder llamar a la función
nodos= strtok(nodos,'.');
```

```
% Ejecutamos la función
met=str2func(nodos);
[t,y]=met();

% Recogemos las abscisas donde aproximar
abscisas=get(handles.edit_datos,'String');
abscisas=fliplr(deblank(fliplr(deblank(abscisas)))); %Eliminamos
huecos en blanco
if isempty(abscisas)
    x=[];
    opx='n';
else
    opx='s';
    % Eliminamos la extensión del archivo para poder llamar a la
función
abscisas= strtok(abscisas,'.');
% Ejecutamos la función
met=str2func(abscisas);
x=met();
end

% Recogemos los pesos
pesos=get(handles.edit_pesos,'String');
pesos=fliplr(deblank(fliplr(deblank(pesos)))); %Eliminamos huecos en
%blanco
% Comprobamos que no está vacío
        if isempty(pesos)==1
            uiwait(msgbox('Debe introducir un valor para los pesos.',
'Mensaje de error',...
                'error','modal'))
            return;
        end

% Eliminamos la extensión del archivo para poder llamar a la función
pesos= strtok(pesos,'.');
% Ejecutamos la función
met=str2func(pesos);
p=met();

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONDICIONES DE CONTORNO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Inicializamos algunas variables por defecto
S1a=0; S2a=0; S1b=0; S2b=0;
CI=0; CD=0;
% Frontera Izquierda
if get(handles.radio_primera_izq,'Value')==1
   CI=1;
   % Obtenemos la primera derivada a la izquierda
        aux = get(handles.edit_primera_izq,'String');
        S1a=sscanf(aux, '%f');
        % Comprobamos que no está vacío
        if isempty(S1a)==1
            uiwait(msgbox('Debe introducir un valor para la primera
derivada en la frontera izquierda.', 'Mensaje de error',...
                'error','modal'))
            return;
        end
```

```
elseif get(handles.radio_segunda_izq,'Value')==1
    CI=2;
   % Obtenemos la primera derivada a la izquierda
        aux = get(handles.edit_segunda_izq,'String');
        S2a=sscanf(aux, '%f');

        % Comprobamos que no está vacío
        if isempty(S2a)==1
            uiwait(msgbox('Debe introducir un valor para la segunda
derivada en la frontera izquierda.', 'Mensaje de error',...
                'error','modal'))
            return;
        end
elseif get(handles.radio_periodica_izq,'Value')==1
    CI=3;
    CD=3;
end

% Frontera derecha
if get(handles.radio_primera_drch,'Value')==1
   CD=1;
   % Obtenemos la primera derivada a la derecha
        aux = get(handles.edit_primera_drch,'String');
        S1b=sscanf(aux, '%f');

        % Comprobamos que no está vacío
        if isempty(S1b)==1
            uiwait(msgbox('Debe introducir un valor para la primera
derivada en la frontera derecha.', 'Mensaje de error',...
                'error','modal'))
            return;
        end

elseif get(handles.radio_segunda_drch,'Value')==1
    CD=2;
   % Obtenemos la primera derivada a la derecha
        aux = get(handles.edit_segunda_drch,'String');
        S2b=sscanf(aux, '%f');

        % Comprobamos que no está vacío
        if isempty(S2b)==1
            uiwait(msgbox('Debe introducir un valor para la segunda
derivada en la frontera derecha.', 'Mensaje de error',...
                'error','modal'))
            return;
        end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% OPCIONES DE GRÁFICAS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


opg1=get(handles.check_valores,'Value');
opg2=get(handles.check_splines,'Value');
opg3=get(handles.check_primera,'Value');
opg4=get(handles.check_segunda,'Value');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% LLAMADA A LA FUNCIÓN QUE CALCULA LOS SPLINES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

cd('./Splines Suavizantes');
Splines_Suavizantes(t',y',CI,S1a,S2a,CD,S1b,S2b,opx,x,opg1,opg2,opg3,o
pg4,p,handles);
cd ..

% --- Executes on button press in push_ficheros.
function push_ficheros_Callback(hObject, eventdata, handles)
% hObject    handle to push_ficheros (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cd('./Splines Suavizantes');
edit polinomios_splines.txt;
edit resul_valores_aproximados.txt;
cd ..


% --- Executes on button press in push_graficas.
function push_graficas_Callback(hObject, eventdata, handles)
% hObject    handle to push_graficas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


%Importamos imagen *.jpg
grafica_inicial=imread('escudo.jpg');
axes(handles.grafica);

%imshow(escudo_upct);
imagesc(grafica_inicial); axis off

% Cerramos las gráficas
allPlots=findall(0,'Type','figure','FileName',[])
delete(allPlots);


% --- Executes on button press in push_reset.
function push_reset_Callback(hObject, eventdata, handles)
% hObject    handle to push_reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Cerramos las gráficas y limpiamos la pantalla
allPlots=findall(0,'Type','figure','FileName',[]);
delete(allPlots);
clc;

%Borramos lo escrito en los recuadros
set(handles.edit_nodos,'String','');
set(handles.edit_datos,'String','');
set(handles.edit_pesos,'String','');
set(handles.edit_primera_izq,'String','');
set(handles.edit_segunda_izq,'String','');
set(handles.edit_primera_drch,'String','');
```

```
set(handles.edit_segunda_drch,'String','');

%Ponemos a cero el panel de condiciones de contorno
set(handles.edit_primera_izq,'Enable','off');
set(handles.edit_segunda_izq,'Enable','off');
set(handles.edit_primera_drch,'Enable','off');
set(handles.edit_segunda_drch,'Enable','off');
set(handles.radio_primera_izq,'Value',0);
set(handles.radio_segunda_izq,'Value',0);
set(handles.radio_periodica_izq,'Value',0);
set(handles.radio_primera_drch,'Value',0);
set(handles.radio_segunda_drch,'Value',0);
set(handles.radio_periodica_drch,'Value',0);


%Ponemos a cero el panel de graficas
set(handles.check_valores,'Value',0);
set(handles.check_splines,'Value',0);
set(handles.check_primera,'Value',0);
set(handles.check_segunda,'Value',0);

% Dibuja la gráfica inicial

%Importamos imagen *.jpg
grafica_inicial=imread('escudo.jpg');
axes(handles.grafica);

imagesc(grafica_inicial); axis off


% --- Executes on button press in push_salir.
function push_salir_Callback(hObject, eventdata, handles)
% hObject    handle to push_salir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
SplinesSuavizantes;


% --- Executes on button press in check_valores.
function check_valores_Callback(hObject, eventdata, handles)
% hObject    handle to check_valores (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of check_valores

% Si no se van a interpolar datos, no se puede marcar su grafica
% correspondiente
aux = get(handles.edit_datos,'String');

if isempty(aux)
    set(handles.check_valores,'Value',0);
end


% --- Executes on button press in check_splines.
function check_splines_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to check_splines (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of check_splines

% --- Executes on button press in check_primera.
function check_primera_Callback(hObject, eventdata, handles)
% hObject    handle to check_primera (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of check_primera

% --- Executes on button press in check_segunda.
function check_segunda_Callback(hObject, eventdata, handles)
% hObject    handle to check_segunda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of check_segunda

% --- Executes on button press in radio_primera_izq.
function radio_primera_izq_Callback(hObject, eventdata, handles)
% hObject    handle to radio_primera_izq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radio_primera_izq

if get(handles.radio_primera_izq,'Value')==1
    set(handles.edit_primera_izq,'Enable','on');
    set(handles.edit_segunda_izq,'Enable','off');
    set(handles.radio_segunda_izq,'Value',0);
    set(handles.radio_periodica_izq,'Value',0);
    set(handles.radio_periodica_drch,'Value',0);
else
    set(handles.edit_primera_izq,'Enable','off');
end


% --- Executes on button press in radio_segunda_izq.
function radio_segunda_izq_Callback(hObject, eventdata, handles)
% hObject    handle to radio_segunda_izq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radio_segunda_izq

if get(handles.radio_segunda_izq,'Value')==1
    set(handles.edit_segunda_izq,'Enable','on');
    set(handles.edit_primera_izq,'Enable','off');
    set(handles.radio_primera_izq,'Value',0);
    set(handles.radio_periodica_izq,'Value',0);
    set(handles.radio_periodica_drch,'Value',0);
else
    set(handles.edit_segunda_izq,'Enable','off');
end
```

```
% --- Executes on button press in radio_periodica_izq.
function radio_periodica_izq_Callback(hObject, eventdata, handles)
% hObject    handle to radio_periodica_izq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of
radio_periodica_izq

if get(handles.radio_periodica_izq,'Value')==1
    set(handles.edit_primera_izq,'Enable','off');
    set(handles.edit_segunda_izq,'Enable','off');
    set(handles.radio_periodica_drch,'Value',1);
    set(handles.radio_primera_izq,'Value',0);
    set(handles.radio_segunda_izq,'Value',0);
    set(handles.edit_primera_drch,'Enable','off');
    set(handles.edit_segunda_drch,'Enable','off');
    set(handles.radio_primera_drch,'Value',0);
    set(handles.radio_segunda_drch,'Value',0);
else
    set(handles.radio_periodica_drch,'Value',0);
end

% --- Executes on button press in radio_primera_drch.
function radio_primera_drch_Callback(hObject, eventdata, handles)
% hObject    handle to radio_primera_drch (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of
radio_primera_drch

if get(handles.radio_primera_drch,'Value')==1
    set(handles.edit_primera_drch,'Enable','on');
    set(handles.edit_segunda_drch,'Enable','off');
    set(handles.radio_segunda_drch,'Value',0);
    set(handles.radio_periodica_drch,'Value',0);
    set(handles.radio_periodica_izq,'Value',0);
else
    set(handles.edit_primera_drch,'Enable','off');
end

% --- Executes on button press in radio_segunda_drch.
function radio_segunda_drch_Callback(hObject, eventdata, handles)
% hObject    handle to radio_segunda_drch (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of
radio_segunda_drch

if get(handles.radio_segunda_drch,'Value')==1
    set(handles.edit_segunda_drch,'Enable','on');
    set(handles.edit_primera_drch,'Enable','off');
    set(handles.radio_primera_drch,'Value',0);
    set(handles.radio_periodica_drch,'Value',0);
    set(handles.radio_periodica_izq,'Value',0);
else
    set(handles.edit_segunda_drch,'Enable','off');
end
```

```
% --- Executes on button press in radio_periodica_drch.
function radio_periodica_drch_Callback(hObject, eventdata, handles)
% hObject    handle to radio_periodica_drch (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of
radio_periodica_drch

if get(handles.radio_periodica_drch,'Value')==1
    set(handles.edit_primera_drch,'Enable','off');
    set(handles.edit_segunda_drch,'Enable','off');
    set(handles.radio_periodica_izq,'Value',1);
    set(handles.radio_primera_drch,'Value',0);
    set(handles.radio_segunda_drch,'Value',0);
    set(handles.edit_primera_izq,'Enable','off');
    set(handles.edit_segunda_izq,'Enable','off');
    set(handles.radio_primera_izq,'Value',0);
    set(handles.radio_segunda_izq,'Value',0);
else
    set(handles.radio_periodica_izq,'Value',0);
end

function edit_primera_izq_Callback(hObject, eventdata, handles)
% hObject    handle to edit_primera_izq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_primera_izq as
text
%        str2double(get(hObject,'String')) returns contents of
edit_primera_izq as a double


% --- Executes during object creation, after setting all properties.
function edit_primera_izq_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_primera_izq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit_primera_drch_Callback(hObject, eventdata, handles)
% hObject    handle to edit_primera_drch (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_primera_drch
as text
%        str2double(get(hObject,'String')) returns contents of
edit_primera_drch as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit_primera_drch_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_primera_drch (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit_segunda_izq_Callback(hObject, eventdata, handles)
% hObject    handle to edit_segunda_izq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_segunda_izq as
text
%        str2double(get(hObject,'String')) returns contents of
edit_segunda_izq as a double


% --- Executes during object creation, after setting all properties.
function edit_segunda_izq_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_segunda_izq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit_segunda_drch_Callback(hObject, eventdata, handles)
% hObject    handle to edit_segunda_drch (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_segunda_drch
as text
%        str2double(get(hObject,'String')) returns contents of
edit_segunda_drch as a double


% --- Executes during object creation, after setting all properties.
function edit_segunda_drch_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_segunda_drch (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit_nodos_Callback(hObject, eventdata, handles)
% hObject    handle to edit_nodos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_nodos as text
%        str2double(get(hObject,'String')) returns contents of
edit_nodos as a double


% --- Executes during object creation, after setting all properties.
function edit_nodos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_nodos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit_datos_Callback(hObject, eventdata, handles)
% hObject    handle to edit_datos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_datos as text
%        str2double(get(hObject,'String')) returns contents of
edit_datos as a double


% --- Executes during object creation, after setting all properties.
function edit_datos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_datos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit_pesos_Callback(hObject, eventdata, handles)
% hObject    handle to edit_pesos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_pesos as text
%        str2double(get(hObject,'String')) returns contents of
edit_pesos as a double


% --- Executes during object creation, after setting all properties.
function edit_pesos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_pesos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushbutton12.
function pushbutton12_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('Tutorial de la Interfaz gráfica.pdf');

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over
pushbutton12.
function pushbutton12_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('Tutorial de la Interfaz gráfica.pdf');
```

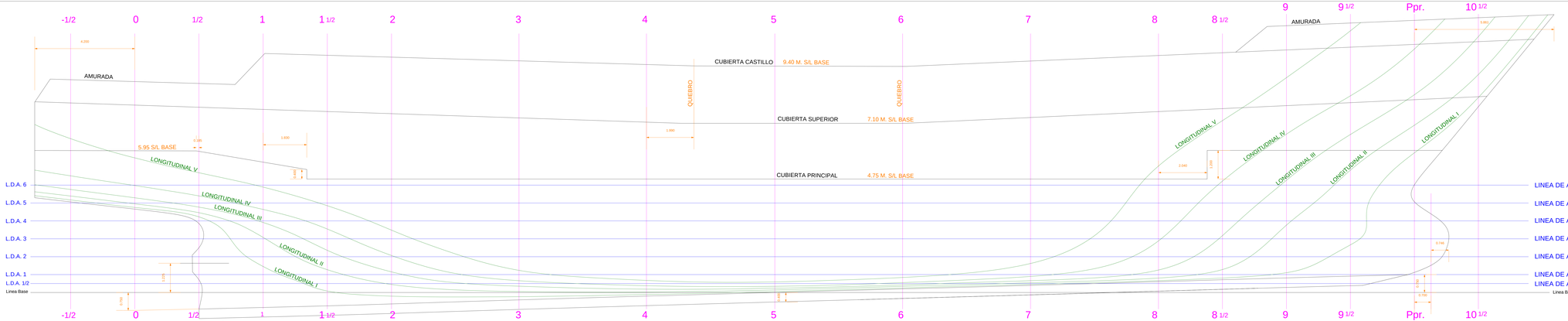
Código Matlab del Menú Ayuda.

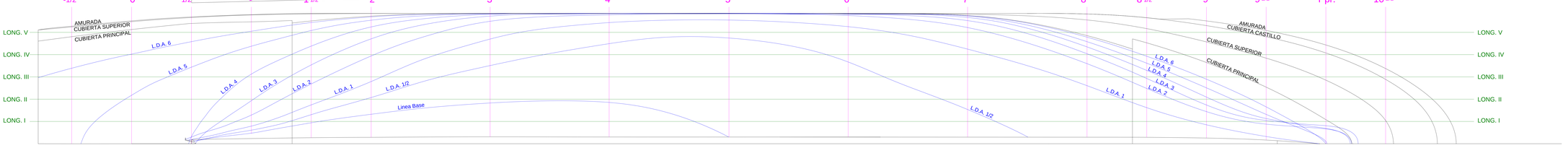
```
function varargout = menu_ayuda(varargin)
% MENU_AYUDA M-file for menu_ayuda.fig
%      MENU_AYUDA, by itself, creates a new MENU_AYUDA or raises the
existing
%      singleton*.
%
%      H = MENU_AYUDA returns the handle to a new MENU_AYUDA or the
handle to
%      the existing singleton*.
%
%      MENU_AYUDA('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in MENU_AYUDA.M with the given input
arguments.
%
%      MENU_AYUDA('Property','Value',...) creates a new MENU_AYUDA or
raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before menu_ayuda_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to menu_ayuda_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help menu_ayuda

% Last Modified by GUIDE v2.5 13-Apr-2013 17:27:28

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @menu_ayuda_OpeningFcn, ...
                   'gui_OutputFcn',  @menu_ayuda_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

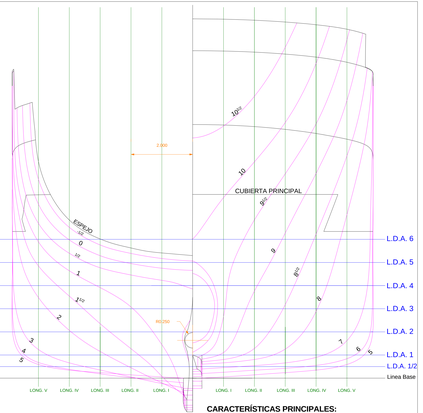
```
% --- Executes just before menu_ayuda is made visible.
function menu_ayuda_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to menu_ayuda (see VARARGIN)

% Choose default command line output for menu_ayuda
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


% UIWAIT makes splines_cubicos_suavizantes wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%% CENTRAR INTERFAZ GRÁFICA EN PANTALLA %%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

set( handles.output,'Units', 'pixels' );
screenSize = get(0, 'ScreenSize');

position = get( handles.output,'Position');
position(1) = (screenSize(3)-position(3))/2;
position(2) = (screenSize(4)-position(4))/2;
set( handles.output,'Position', position );
%%%Introducimos el fondo con el escudo de la UPCT%%%
% create an axes that spans the whole gui
ah = axes('unit', 'normalized', 'position', [0 0 1 1]);
% import the background image and show it on the axes
bg = imread('fondo_principal.jpg'); imagesc(bg);
% prevent plotting over the background and turn the axis off
set(ah,'handlevisibility','off','visible','off')
% making sure the background is behind all the other uicontrols
uistack(ah, 'bottom');


% --- Outputs from this function are returned to the command line.
function varargout = menu_ayuda_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
ah = axes('unit', 'normalized', 'position', [0 0 1 1]);
```

```
% prevent plotting over the background and turn the axis off
set(ah,'handlevisibility','off','visible','off')
% making sure the background is behind all the other uicontrols
uistack(ah, 'bottom');
% create an axes that spans the whole gui
ah = axes('unit', 'normalized', 'position', [0 0 1 1]);
% import the background image and show it on the axes
bg = imread('fondo_principal.jpg'); imagesc(bg);
% prevent plotting over the background and turn the axis off
set(ah,'handlevisibility','off','visible','off')
% making sure the background is behind all the other uicontrols
uistack(ah, 'bottom');

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
SplinesSuavizantes;
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on mouse press over figure background.
function figure1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('Splines cubicos suavizantes.pdf');

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('Tutorial de la Interfaz gráfica.pdf');


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('Resolución de Splines en Matlab.pdf');
```

Anexo III. Plano de Formas.







CARACTERÍSTICAS PRINCIPALES:

-Eslora entre perpendiculares.....53.700 m
-Asiento de proyecto.....1.500 m
-Manga de trazado.....11.700 m
-Puntal cbta. superior.....7.100 m
-Puntal cbta. principal.....4.750 m
-Calado de trazado.....4.500 m
-Separación entre lineas de agua.....750 mm
-Separación entre secciones.....5370 mm
-Separación entre longitudinales.....1000 mm

Universidad Politécnica de Cartagena

BUQUE ARRASTRERO

PLANO DE FORMAS

ESCALA	INGENIERÍA NAVAL Y OCEÁNICA		
1:50	FECHA: 20/07/2013		
	PLANO Nº: 1	Nº DE HOJAS: 1	

Blanca Roncero Peña

Bibliografía

- Apuntes de Métodos Numéricos.

  Juan Carlos Trillo Moya. Universidad Politécnica de Cartagena (2012).

- Curso de Matemáticas Superiores.

  M.L. Krasnov. A.I. Kiseliov. G.I. Makárenko. Ie. V. Shikin. V.I. Zialiapin. Editorial URSS, Moscú (2003).

- Curves and Surfaces for CAGD: a Practical Guide. 5ª Edición.

  G. Farin. Morgan Kaufmann Publishers, San Francisco (2002).

- Computer Graphics and Geometric Modeling.

  D. Salomon. Springer Verlag, New York (1999).

- NURBS: from Projective Geometry to Practical Use. 2 edición.

  G. Farin. AK Peters Ltd., Natick (1999).

- The NURBS Book. 2 edición.

  L. Piegl, W. Tiller. Springer Verlag, Berlin (1997).

- Fundamentals of Computer Aided Geometric Design.

  J. Hoschek, D. Lasser. AK Peters Ltd., Wellesley (1993).

- Mathematical Elements for Computer Graphics.

  D.F. Rogers, J .A. Adams. McGraw-Hill, New York (1990).

- A Practical Guide to Splines.

  C. de Boor. Springer Verlag, New York (1978).

- Splines Cúbicos Interpolantes en el Diseño Naval.

  Irene Gallego Valdellós. Universidad Politécnica de Cartagena. Cartagena (2012).

- Apuntes de Sistemas de Construcción de Buques y Artefctos.

  Alfonso Martínez García. Universidad Politécnica de Cartagena. Cartagena (2012).

- Construcción Naval III

  Chorro Oncina, Rosendo. ETSIN Universidad Politécnica de Madrid. Madrid (1960).

- Técnicas de Construcción Naval. 2ª Edición.

  González López, Primitivo B. Universidad da Coruña. La Coruña (2005).

Las siguientes páginas web fueron consultadas en el periodo comprendido entre Enero de 2013 y Junio de 2013:

- www.wikipedia.org